TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Betriebswissenschaften und Montagetechnik
am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*)

Automatische Konfiguration von Robotersystemen (Plug&Produce)

Stefan Alexander Krug

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr.-Ing. Michael Zäh
Prüfer der Dissertation:	1. Univ.-Prof. Dr.-Ing. Gunther Reinhart 2. Univ.-Prof. Dr.-Ing. Jörg Krüger Technische Universität Berlin

Die Dissertation wurde am 20.06.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 14.11.2012 angenommen.

– Um große Ziele zu erreichen, bedarf es zweierlei:
Geist und Begeisterung. –

Geleitwort der Herausgeber

Die Produktionstechnik ist für die Weiterentwicklung unserer Industriegesellschaft von zentraler Bedeutung, denn die Leistungsfähigkeit eines Industriebetriebes hängt entscheidend von den eingesetzten Produktionsmitteln, den angewandten Produktionsverfahren und der eingeführten Produktionsorganisation ab. Erst das optimale Zusammenspiel von Mensch, Organisation und Technik erlaubt es, alle Potentiale für den Unternehmenserfolg auszuschöpfen.

Um in dem Spannungsfeld Komplexität, Kosten, Zeit und Qualität bestehen zu können, müssen Produktionsstrukturen ständig neu überdacht und weiterentwickelt werden. Dabei ist es notwendig, die Komplexität von Produkten, Produktionsabläufen und -systemen einerseits zu verringern und andererseits besser zu beherrschen.

Ziel der Forschungsarbeiten des *iwb* ist die ständige Verbesserung von Produktentwicklungs- und Planungssystemen, von Herstellverfahren sowie von Produktionsanlagen. Betriebsorganisation, Produktions- und Arbeitsstrukturen sowie Systeme zur Auftragsabwicklung werden unter besonderer Berücksichtigung mitarbeiterorientierter Anforderungen entwickelt. Die dabei notwendige Steigerung des Automatisierungsgrades darf jedoch nicht zu einer Verfestigung arbeitsteiliger Strukturen führen. Fragen der optimalen Einbindung des Menschen in den Produktentstehungsprozess spielen deshalb eine sehr wichtige Rolle.

Die im Rahmen dieser Buchreihe erscheinenden Bände stammen thematisch aus den Forschungsbereichen des *iwb*. Diese reichen von der Entwicklung von Produktionssystemen über deren Planung bis hin zu den eingesetzten Technologien in den Bereichen Fertigung und Montage. Steuerung und Betrieb von Produktionssystemen, Qualitätssicherung, Verfügbarkeit und Autonomie sind Querschnittsthemen hierfür. In den *iwb* Forschungsberichten werden neue Ergebnisse und Erkenntnisse aus der praxisnahen Forschung des *iwb* veröffentlicht. Diese Buchreihe soll dazu beitragen, den Wissenstransfer zwischen dem Hochschulbereich und dem Anwender in der Praxis zu verbessern.

Gunther Reinhart Michael Zäh

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) der Technischen Universität München.

Herrn Professor Gunther Reinhart und Herrn Professor Michael Zäh, den Leitern dieses Instituts, gilt mein besonderer Dank für die Möglichkeit der Promotion sowie für die wohlwollende Förderung und großzügige Unterstützung meiner Arbeit. Herrn Professor Krüger (Technische Universität Berlin) danke ich sehr herzlich für die Übernahme des Korreferates und die aufmerksame Durchsicht dieser Arbeit.

Darüber hinaus bedanke ich mich bei den Kollegen, die mir in den vielen Stunden der Diskussion mit fachlichem und freundschaftlichem Rat zur Seite gestanden sind und damit die Zeit am Institut unvergesslich gemacht haben. Besonders hervorheben möchte ich Herrn Braunreuther, Herrn Egbers, Herrn Mari, Herrn Meling und Herrn Ulrich, die meine Arbeit durch ihre kritische Durchsicht mit wertvollen Anregungen bereichert haben.

Bei Herrn Mari, Herrn Schlögel, Herrn Bauer, Herrn Schmidt und insbesondere bei Herrn Hammerstingl sowie Herrn Braun, die meine Forschung im Rahmen ihres Studiums begleitet haben, bedanke ich mich für ihr Engagement.

Meinen Eltern, meiner Schwester und meinen Freunden, die immer an mich geglaubt haben, danke ich für den nötigen Rückhalt beim Erstellen dieser Arbeit.

In ganz besonderer Weise möchte ich mich bei meiner Frau Lena für ihre Geduld, Nachsicht und die immerwährende Unterstützung bedanken. Sie hat mir währen der gesamten Zeit der Promotion stets den Rücken frei gehalten. Ihr sei dieses Buch gewidmet.

München, im Dezember 2012 Stefan Krug

Inhaltsverzeichnis

Verzeichnis der Abkürzungen und Akronyme

3D	dreidimensional
A	Ampere
ACFG	Autoconfiguration Management Framework für Feldbusse
AG	Aktiengesellschaft
AO	Architektur Objekt
API	Application Programming Interface
ARIKT	Automatische roboterbasierte Inspektion komplexer Teile
ASCII	American Standard Code for Information Interchange
AutomationML	Automation Modeling Language
Beschr.	Beschreibung
BMBF	Bundesministerium für Bildung und Forschung
bspw.	beispielsweise
bzgl.	bezüglich
bzw.	beziehungsweise
ca.	cica
CA	Collision Avoidance
CAD	Computer Aided Design
CAEX	Computer Aided Engineering Exchange
CAN	Controller Area Network
CCC	Computing Community Consortium
CD	Collision Detection
CN	Controlled Node
COLLADA™	Collaborative Design Activity

CSMA	Carrier Sence Multiple Access
DHCP	Dynamic Host Configuration Protocol
DIN	Deutsches Institut für Normung
div.	diverse
DKP	Dreh-Kipp-Positionierer
DLL	Dynamic Link Library
E/A	Ein- und Ausgänge
EEDD	erweiterte Gerätebeschreibung
EDDL	Electronic Device Description Language
EL	Elektronik
EmsA	Entwicklungssystem für modulare, selbstkonfigurierende Visualisierungen zur Anlagenüberwachung
EPSG	European Powerlink Standardization Group
ERP	Enterprise Resource Planning
et al.	et alii
etc.	et cetera
ext.	extern
FCS	Frame Check Sequence
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GENA	General Event Notification Architecture
Geometr.	Geometrie
ggf.	gegebenenfalls
GmbH	Gesellschaft mit beschränkter Haftung
GSD	Gerätestammdatei

GUI	Graphical User Interface
HDR	Header
Hrsg.	Herausgeber
HTTP	Hypertext Transfer Protocol
HÜMNOS	Herstellerübergreifende Module für den nutzerorientierten Einsatz der offenen Steuerungsarchitektur
HW	Hardware
I*PROMS	Inovative Production Machines and Systems
ID	Indentifikationsnummer
IEC	International Electrotechnical Commission
IEM	Industrial Ethernet Modul
IFR	International Federation of Robotics
IG	Informationsgewinnung
IGES	Initial Graphics Exchange Specification
IP	Internet Protocol
IRL	Industrial Robot Language
ISO	International Organiziation for Standardization
IT	Informationstechnik
JRA	Japan Robot Association
kB	Kilobyte
K	Konfiguration
KRL	Kuka Robot Language
LAN	Local Area Network
LS	Liniensteuerung
MAC	Media Access Control

MB	Megabyte
MES	Manufacturing Execution System
MN	Managing Node
ms	Millisekunde
NC	Numerische Steuerung
Nr.	Nummer
OLE	Object Linking and Embedding
OLP	Offline-Programmierung
OPC	OLE for Process Control
ORiN	Open Resource Interface for the Network
OSACA	Open System Architecture for Controls within Automation Systems
OSI	Open System Interconnection
PAPAS	Plug and Play Antriebs- und Steuerungstechnik
PC	Personal Computer
PDO	Process Data Object
PHY	Physikalische Schicht
RC	Robotersteuerung
s	Sekunde
S.	Seite
Schnittst.	Schnittstelle
SD	Secure Disk
SDO	Service Data Object
SIARAS	Skill-based Inspection and Assembly of Reconfigurable Automation Systems

SOAP	Simple Object Access Protocol
SPS	Speicherprogrammierbare Steuerung
SSDP	Simple Service Discovery Protocol
STEP	Standard for the Exchange of Product Model Data
SW	Software
SysML	Systems Modeling Language
TCP	Transmission Control Protocol
TN	Teilnehmer
u. a.	unter anderem
UDP	User Datagram Protocol
UML	Unified Modeling Language
UPnP™	Universal Plug and Play
USB	Universal Serial Bus
V	Volt
VDI	Verein Deutscher Ingenieure
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
vgl.	vergleiche
VP	Versuchsperson
VRML	Virtual Reality Modeling Language
XDD	XML Device Description
XIRP	XML Interface für Roboter und Peripheriegeräte
XML	Extensible Markup Language
z. B.	zum Beispiel

1 Einleitung

1.1 Trends in der industriellen Robotik

Roboter nehmen in vielen Industriestaaten einen wichtigen Stellenwert innerhalb der Produktion ein. Sie steigern die Produktivität und tragen entscheidend zur Verbesserung der Produkt- und Prozessqualität bei. Durch neue Steuerungsgenerationen, höhere Genauigkeit und vor allem durch die Integration von Sensoren konnten in den letzten Jahren neue Prozesse und Anwendungen für Industrieroboter etabliert werden. Heute fertigen diese flexiblen Produktionssysteme in hoch automatisierten Anlagen meist Großserien mit mehreren Tausend Stück. (Reinhart & Krug 2009)

Bei geringen Stückzahlen ist der Einsatz von Industrierobotern hingegen zurückhaltend. Nur 22 % der Unternehmen setzen Roboter in der Kleinserien- und 14 % in der Einzelfertigung ein (Armbruster et al. 2006). Gründe für die geringe Verbreitung in diesem Umfeld sind zum einen fehlende technische Lösungen und zum anderen die hohen Kosten für automatisierte Robotersysteme, die nur zu einem geringen Teil aus den Anschaffungskosten des Roboters an sich bestehen (Zäh et al. 2004). So sind nach Bartscher (2011) die Integrationskosten eines Industrieroboters in eine Automatisierungslösung bis zu zehnmal höher als die Kosten für den Roboter selbst.

Die produktionsrelevanten Megatrends (vgl. Abele & Reinhart 2011) werden die Einsatzbedingungen für Industrieroboter verschärfen. Eine Betrachtung dieser Trends zeichnet ein Zukunftsbild der Automatisierungstechnik mit veränderten Randbedingungen: Beschleunigte Entwicklung, individuelle Produkte, eine hohe Variantenvielfalt und unsichere Prognosen werden die Produktion von morgen prägen (Abele & Reinhart 2011). Die Anzahl kleiner Losgrößen wird weiter zunehmen und damit werden neue Anforderungen an Produktionsanlagen hinsichtlich Flexibilität und Wandlungsfähigkeit gestellt (Spath & Scholtz 2007; Hedelind & Jackson 2007).

Robotersysteme sind derzeit noch nicht für diese sich verändernden Anforderungen ausgelegt. Zur Verdeutlichung der Hemmnisse des Robotereinsatzes in der Kleinserie lohnt eine Betrachtung der konventionellen Prozesskette zur

Erstellung einer Roboteranwendung: Die wesentlichen Schritte sind die Planung, die Systemintegration und der Produktionsbetrieb.

Die Planung erfolgt heute überwiegend am Computer. Mithilfe von CAD-Programmen werden das Layout einer Zelle geplant, mechanische Komponenten entworfen und Haltevorrichtungen konstruiert. Im Rahmen der sogenannten Systemintegration werden der Roboter sowie die einzelnen Komponenten aufgebaut und montiert. Dies beinhaltet neben der physikalischen Signalverbindung und der Konfiguration der Datenschnittstellen zwischen Roboter und Peripheriegeräten auch die Interpretation der Daten an den jeweiligen Systemen sowie die Erstellung gerätespezifischer Funktionen. Die Programmierung erfolgt meist offline mit einem CAD-Modell der Roboterzelle oder online direkt am Roboter. Bei der Inbetriebnahme wird schließlich das System getestet und in den Produktionsbetrieb überführt.

Aus dieser Prozesskette ergeben sich Konsequenzen für den Robotereinsatz: Durch die komplexer werdenden Systeme induziert, werden Experten für die Automatisierungstechnik benötigt, die nicht nur mit dem durchzuführenden Prozess vertraut sind, sondern auch Erfahrung mit der erforderlichen Automatisierungs-, Geräte- und Robotertechnik haben. Die Zeit für die Einrichtung und Programmierung eines Robotersystems muss kurz sein, da bei Kleinserien das Verhältnis von Einrichtzeit zu Produktionsbetrieb sonst unwirtschaftlich wird. Erhebliche Konfigurationsaufwände entstehen dabei durch die große Vielfalt des Funktionsumfangs der Geräte und die herstellerspezifische Schnittstellengestaltung.

Neben einer vereinfachten Programmierung kann vor allem die Reduktion des Konfigurationsaufwands einen wesentlichen Beitrag zur Verkürzung der Aufbauzeit sowie der Inbetriebnahme von Robotersystemen leisten und diese damit flexibler gestalten. So werden Industrieroboter für den zukünftigen Einsatz in einem sich ständig ändernden Umfeld qualifiziert. (Hedelind & Jackson 2007)

1.2 Ausgangssituation

Der wachsende Bedarf an flexiblen und wandlungsfähigen Robotersystemen führt zu steigenden Anforderungen an deren Konfiguration und Rekonfiguration

(Reinhart & Krug 2009). Gründe dafür sind maßgeblich produktinduzierte Veränderungen der Produktionsanforderungen, die Herstellung neuer Produkte auf bestehenden Anlagen und die Integration neuer Prozesse oder Technologien in bestehende Produktionsstrukturen (I*PROMS 2006).
Nicht nur technische, sondern auch wirtschaftliche Gründe können einen Konfigurationsbedarf auslösen. Eingesetzte Geräte und Roboter müssen in Zukunft für andere Aufgaben weiterverwendet werden, da sich die Amortisation der gerätetechnischen Investition vor allem bei Kleinserien nicht in einem Produktionseinsatz einstellen kann. Zunehmend wird also eine mehrmalige Veränderung des gerätetechnischen Aufbaus einer Roboterzelle während ihrer Lebensdauer erforderlich. Diese Veränderung ist meist mit einem großen manuellen Aufwand verbunden. Besonders kleine und mittlere Unternehmen stellt dies vor große Herausforderungen.

Der grundlegende Bedarf an derartigen, einfach umzurüstenden Industrieroboter wird durch die Forschungsziele des Weißbuches „Industrial Robot Automation“ (Hägele et al. 2005) und der „Roadmap for US Robotics“ (CCC 2009) betont. Darin wird die Fähigkeit zur einfachen Umrüstung von Industrieroboter als ein Kernziel der Roboterforschung und -entwicklung dargestellt.

Derzeitige Entwicklungen in der Automatisierungstechnik schaffen geeignete Rahmenbedingungen für neue Konzepte rekonfigurierbarer Robotersysteme:

Modularität wird als eine der wichtigsten Eigenschaften hinsichtlich Wandlungsfähigkeit gesehen. Peripheriegeräte, auch auf Sensor-Aktor-Ebene, werden verstärkt befähigt, Funktionen selbstständig auszuführen. Die Modularisierung der Geräte wird zunehmen (Harbach et al. 2007).

Verteilte Steuerungsarchitekturen unterstützen die Modularität. Informations-, Kommunikations- und Steuerungsfähigkeiten werden in die Geräte verlagert und bilden damit mechatronisch abgeschlossene Module, welche in die Gesamtanlage integriert werden (Fussel 2000).

Ethernet-basierte Bussysteme nehmen in ihrer Verbreitung stetig zu. Nach Kuppinger (2004) setzen bereits 70 % der Industrieanwender Ethernet-basierte Busse ein. Mit Echtzeit-Ethernet-Netzwerken besteht die Möglichkeit neben Prozessdaten weitere Informationen über dasselbe Netz mit gängigen Protokollen, wie TCP/IP oder UDP/IP, zu übertragen (Jäger 2009).

Die Konfiguration ist eine Herausforderung mechatronischer Schnittstellen. Dies beinhaltet, dass mechanische, elektrische und informationstechnische Schnittstellen bei der Betrachtung berücksichtigt werden müssen. Insbesondere die mechanischen und elektrischen Schnittstellen sind für heutige Roboter teilweise standardisiert (vgl. z. B. DIN 11593). Dennoch spiegelt sich die Modularisierung der mechanischen und elektrischen Komponenten nicht in der Architektur der Robotersteuerung wieder. Um eine weitestgehend automatische Konfiguration von Robotersystemen zu erreichen, werden neben einer Vorgehensweise zur Modularisierung, neuartige Konzepte für die automatische Konfiguration von Robotersystemen benötigt. Diese sollen, unter Einbeziehung aktueller Entwicklungen der Automatisierungstechnik, eine einfache informationstechnische Integration von Peripheriekomponenten ermöglichen, welche bisher dem System nicht bekannt sind.

1.3 Zielsetzung

Ziel dieser Arbeit ist die Entwicklung einer neuartigen Methode zur automatischen Konfiguration von Robotersystemen (Plug&Produce). Sie soll dazu beitragen, die wirtschaftliche Flexibilität und Wandlungsfähigkeit von Robotersystemen durch eine automatische Integration von Peripheriegeräten zu erhöhen. Die maßgebenden Zielgrößen sind hierbei sowohl die Reduktion des erforderlichen Konfigurationsaufwands als auch des nötigen Expertenwissens zur Anpassung der informationstechnischen Schnittstellen und zur Vorbereitung des Produktionsbetriebs.

Die Methode fokussiert dabei auf die automatische Konfiguration folgender Systemkomponenten unter Verwendung von Industrial-Ethernet-Netzwerken: Die Robotersteuerung soll befähigt werden mit den Peripheriegeräten auf funktionaler Ebene zu kommunizieren und die dazugehörigen Prozessdaten über die eingesetzten Netzwerke auszutauschen. Für Programmierumgebungen (z. B. für Programme zur simulationsgestützten Offline-Programmierung) sollen bspw. entsprechende Projektdateien angelegt werden, die den Initialaufwand zur Nachbildung des Robotersystems reduzieren. Schließlich soll die Erstellung der Dokumentation der Anlage durch die automatische Konfiguration erleichtert und beschleunigt werden.

Basierend auf leistungsfähigen Industrial-Ethernet-Netzwerken, die zunehmend breite Anwendung in der Industrie finden, werden neuartige Prinzipien aufgezeigt, welche eine Plug&Produce-Integration für Roboter und die von Robotern gesteuerten Peripheriegeräte im industriellen Umfeld ermöglichen. Abbildung 1 zeigt die Einordnung des Betrachtungsraumes im Kontext der Automatisierungspyramide nach IEC 62264.

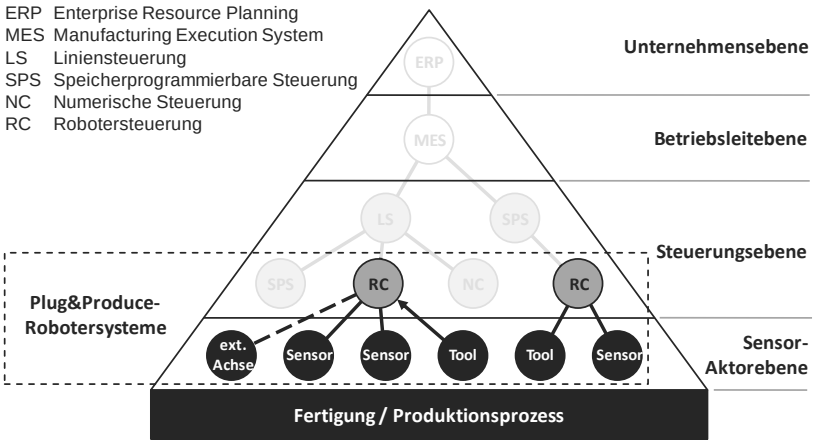

Abbildung 1: Betrachtungsraum innerhalb der Automatisierungspyramide – in Anlehnung an IEC *62264*

1.4 Aufbau der Arbeit

Zur Zielerreichung werden in der vorliegenden Arbeit zunächst die für das Verständnis erforderlichen Grundlagen *(Kapitel 2 Grundlagen)* erläutert. Dies beinhaltet die Struktur und die systemischen Ausprägungen von Industrierobotern und Robotersystemen sowie die Darstellung der Grundlagen zur industriellen Kommunikation. Durch die Erläuterung konventioneller Vorgehensweisen von der Planung bis zur Inbetriebnahme von Robotersystemen wird ein grundlegendes Verständnis für die Hintergründe der Konfiguration geschaffen.

Bestehende Erkenntnisse aus dem PC-Bereich und dem produktionstechnischen Umfeld zur Vereinfachung von Konfigurationsvorgängen werden in *Kapitel 3 Stand der Technik und Forschung* diskutiert und im Kontext kleiner Losgrößen

und steigender Flexibilität und Wandlungsfähigkeit bewertet. Darauf aufbauend werden Konfigurationsbedarfe herausgestellt und der Handlungsbedarf ermittelt.

Das *Kapitel 4 Anforderungsanalyse* ist zweigeteilt. Der derzeitige Kenntnisstand der Forschung erlaubt keine Bestimmung der Informationen, die während eines Konfigurationsvorganges in einer Roboterzelle ausgetauscht werden. Aus diesem Grund wurde zunächst eine Informationsflussanalyse durchgeführt und Rückschlüsse auf das notwendige Spektrum an Informationen sowie auf die Erfordernisse hinsichtlich Datenverfügbarkeit und Konfigurationsprozess gezogen. Auf dieser Grundlage werden schließlich die technischen, wirtschaftlichen sowie nutzerorientierten Anforderungen und Voraussetzungen an Plug&Produce-Robotersysteme abgeleitet.

Eine dieser Voraussetzungen ist die Modularität der Peripheriekomponenten. Ausgehend von der allgemeinen Systemtheorie wird in *Kapitel 5 Modularisierung* ein Konzept zur funktionsorientierten Modularisierung abgeleitet und ein Vorgehen zur Gestaltung funktionaler Plug&Produce-Module vorgestellt.

Den Kern der Arbeit stellt *Kapitel 6 Methode zur automatischen Konfiguration* dar. Hier werden zunächst wichtige Vorüberlegungen zur Methode hinsichtlich Informationsverarbeitung, möglichen Kommunikationssystemen und des Informationsangebots und -nachfrage diskutiert. Darauf aufbauend werden das Wirkprinzip und die Funktionsweise der Methode erläutert, das Vorgehen bei der Informationsverarbeitung beschrieben sowie der Methodenablauf bei der Konfiguration erklärt. Passend zu der entwickelten Methode wird eine softwaretechnische Referenzarchitektur vorgestellt.

Die Umsetzung der Methode sowie die praxisnahe Validierung erfolgt in *Kapitel 7 Experimentelle Umsetzung und Validierung*. Zunächst wird die Versuchsumgebung vorgestellt und deren Peripheriegeräte entsprechend des erarbeiteten Vorgehens zur Modularisierung systemtechnisch strukturiert. Der Konfigurationsmanager wird anhand der Referenzarchitektur implementiert. Schließlich wird mit den modularisierten Komponenten und dem Konfigurationsmanager exemplarisch der automatische Konfigurationsprozess getestet.

In *Kapitel 8 Technische und wirtschaftliche Bewertung* wird die neue Methode der konventionellen Vorgehensweise gegenübergestellt und die Grenzen des Einsatzspektrums erörtert.

Die Arbeit wird durch *Kapitel 9 Zusammenfassung und Schlussbetrachtung* rekapituliert und durch einen Ausblick auf die industrielle und methodische Weiterentwicklung der Technologie beschlossen.

2 Grundlagen

2.1 Allgemeines

Das Kapitel *Grundlagen* beinhaltet neben typischen industriellen Anwendungsgebieten und gängigen Applikationen die wesentlichen Typen von Robotern, deren Aufbau und Programmierung. Des Weiteren wird die Struktur von Robotersystemen dargestellt. Die Kategorien und Eigenschaften von Peripheriegeräten werden beschrieben. Die Grundlagen industrieller Kommunikation werden dargestellt und die Ausprägungen industrieller Netzwerke aufgezeigt. Schließlich wird der konventionelle Vorgang zur Konfiguration eines Robotersystems dargestellt.

2.2 Grundlagen Industrieroboter

2.2.1 Begriffsdefinition

Der Begriff des Industrieroboters wird in verschiedenen Ländern unterschiedlich aufgefasst. In den Industriestaaten weitverbreitete Definitionen sind in der DIN 8373 (1996) und der VDI-Richtlinie 2860 (1990) enthalten.

„Automatisch gesteuerter, frei programmierbarer Mehrzweck-Manipulator, der in drei oder mehr Achsen programmierbar ist und zur Verwendung in der Automatisierungstechnik entweder an einem festen Ort oder beweglich angeordnet sein kann.“ (DIN 8373 1996)

„Industrieroboter sind universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegung hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln frei (d.h. ohne mechanischen Eingriff) programmierbar und ggf. sensorgeführt sind.“ (VDI-Richtlinie 2860)

2.2.2 Applikationen

Die Definition eines Industrieroboters als universelles Handhabungsgerät lässt erahnen, dass die Einsatzgebiete der Roboter grundsätzlich nicht beschränkt sind. Heute finden Industrieroboter in einer Vielzahl von Applikationen Verwendung, die unterschiedliche Anforderungen an die Robotersysteme stellen. Neben den größten Anwendungsgebieten wie Handhabung, Schweißen und Montage gibt es eine Vielfalt an weiteren spezifischen Anwendungen, in welchen Industrieroboter zum Einsatz kommen (vgl. Abbildung 2) (IFR 2011).

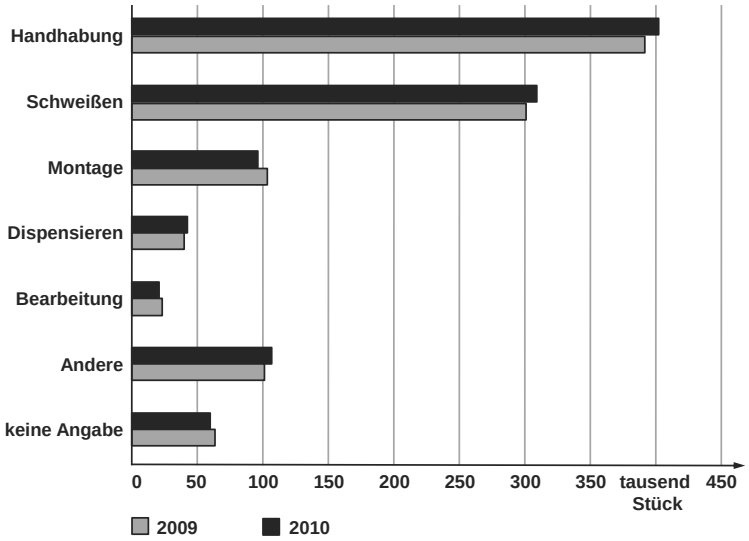

*Abbildung 2: Anzahl der weltweit in Betrieb befindlichen Industrieroboter nach Einsatzgebieten (*IFR *2011)*

In jedem Fall werden Robotersysteme speziell für die zu erledigende Aufgabe ausgewählt und mit entsprechenden Sensoren und Aktoren erweitert, um die gestellten Anforderungen bezüglich Geschwindigkeit, Genauigkeit, Qualität etc. zu erfüllen. So ist eine Roboterzelle üblicherweise das Ergebnis einer kundenindividuellen Planung, Integration, Programmierung und Konfiguration, welches bei der Erstellung ein beträchtliches Expertenwissen erfordert (Siciliano & Khatib 2008).

2.2.3 Einsatzumfeld und Hintergründe

Industrieroboter in der heutigen Form sind maßgeblich durch die Anforderungen der meist kapitalintensiven Massenproduktion – vornehmlich der Automobilindustrie – geprägt, die den Markt auch heute noch dominiert (IFR 2011; Siciliano & Khatib 2008). Robotersysteme sind daher für einen Einsatz in einem strukturierten Umfeld und unter definierten, sich nicht wesentlich ändernden Rahmenbedingungen ausgelegt.

Der anhaltende Preisverfall der Industrieroboter, um ca. 40 % in den letzten 17 Jahren (IFR 2011), führt zu einem steigenden Einsatz von Roboterlösungen bei kleinen und mittleren Unternehmen (Verl & Naumann 2008a). So setzen nach einer Studie von Armbruster et al. (2006) heute mehr als die Hälfte der Unternehmen mit einer Größe von über 250 Mitarbeitern Roboter ein. Bei Betrieben mit einer Größe bis 50 Mitarbeiter sind es noch 12 %. Maßgebliche Hinderungsgründe für einen breiteren Robotereinsatz sind neben der Programmierung die Aufwände und das Engineering für die Erstellung neuer Roboterapplikationen. Diese führen auch bei kleinen Stückzahlen zu einem ungünstigeren Verhältnis von Einrichtzeiten zu Produktionsbetrieb, was eine wirtschaftliche Produktion verhindert. Aufwände sind insbesondere das erforderliche automatisierungstechnische Expertenwissen, der Zeitaufwand, die fehlende Modularität bzw. Flexibilität und die Wiederverwendbarkeit der Roboterkomponenten sowie die Investitionen, welche u. a. mit der Beauftragung externer Systemintegratoren einhergehen (Hedelind & Jackson 2007).

2.2.4 Aufbau und Programmierung von Industrierobotern

Der Roboter besteht nach DIN 8373 (1996) aus zwei wesentlichen Bestandteilen: dem *Manipulator* zur Erzeugung einer Bewegung, und der *Steuerung*, die sowohl die Bewegungen des Manipulators vorgibt bzw. überwacht, als auch die Kommunikation mit der Umgebung (Anlagen und Geräte) ermöglicht (IFR 2011). Unter dem Begriff der *freien Programmierung* (vgl. Definition Industrieroboter Abschnitt 2.1.1) wird die Fähigkeit verstanden, die programmierten Bewegungen oder zusätzlichen Funktionen ohne Modifikationen an der mechanischen Struktur oder an der Steuerung verändern zu können.

Manipulator

Der Manipulator setzt sich aus einer spezifischen Kombination von angetriebenen Achsen und Strukturelementen zusammen. Deren Ausprägung und geometrische Anordnung definieren die mathematische Bewegungsbeschreibung des Roboters, die sogenannte Kinematik. Das Koordinatensystem, welches den Anfang der kinematischen Kette beschreibt, wird als Basis bezeichnet, das Ende als mechanische Schnittstelle (vgl. Abbildung 3) (DIN 8373 1996).

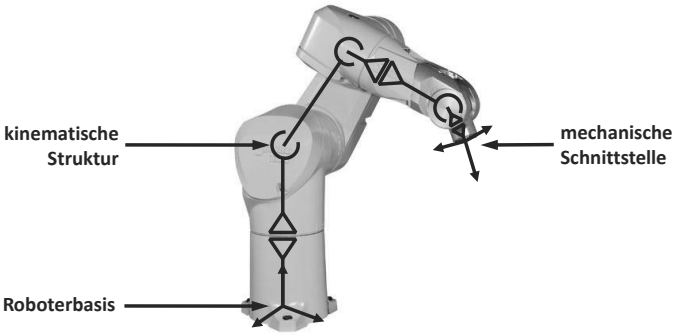

Abbildung 3: Manipulator mit Achsen und kinematischer Beschreibung – Hintergrundbild: Stäubli Tec-Systems GmbH

Je nach Aufbau des Manipulators lassen sich die resultierenden Strukturen in parallele und serielle Kinematiken unterteilen (Husty et al. 1997). Die Achsen serieller Kinematiken bauen aufeinander auf. Wird bspw. die erste Achse bewegt, ändert sich der mechanische Angriffspunkt der zweiten Achse. Bei parallelen Kinematiken existieren mehrere parallele Verbindungen von der Basis zur mechanischen Schnittstelle. Die Pose beschreibt die aktuelle Stellung des Roboters und resultiert aus der Summe der Achsstellungen. (Nof 1999)

Steuerung

Die originären Aufgaben der Robotersteuerung sind die Bahnplanung, also die Berechnung der Robotertrajektorie, die Bahnsteuerung und die damit verbundene Ansteuerung der einzelnen Motoren der Achsen. Des Weiteren ist sie für das Lesen und Schreiben digitaler bzw. analoger Ein- und Ausgänge (E/A)

anhand zuvor erstellter Applikationsprogramme zuständig. Diese Programme enthalten zwei verschiedene Instruktionstypen: Bewegungsanweisungen, die den Roboter entlang einer vorgesehenen Bahn bewegen, und Steuerungsanweisungen, die Ein- bzw. Ausgangssignale lesen oder schreiben und so mit den in der Roboterzelle vorhandenen Peripheriegeräten kommunizieren. Diese Steuerungsanweisungen für und von externen Geräten können durch das individuelle Setzen oder Lesen nummerierter Ein- und Ausgänge programmiert werden. Durch bspw. ein Programmierhandgerät kann der Bediener die verschiedenen Funktionen der Robotersteuerung anwählen und einstellen. (Nof 1999)
In Abbildung 4 ist die grundlegende Funktionsweise einer Robotersteuerung schematisch dargestellt.

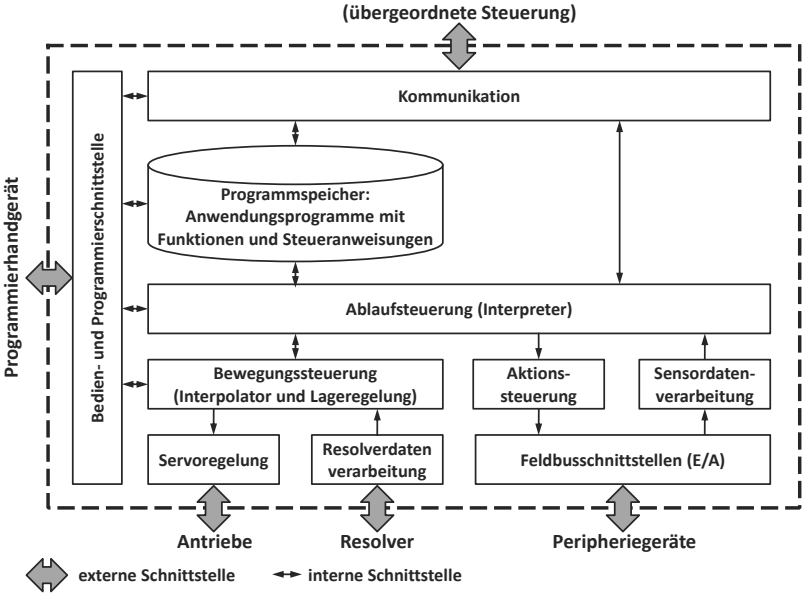

Abbildung 4: Schematischer Aufbau einer Industrierobotersteuerung in Anlehnung an Siciliano & Khatib *(2008),* Weber *(2009) und* Dubbel *(2007)*

Beim Aufruf eines Roboterprogramms wird dieses zunächst aus dem Programmspeicher geladen und von einem Interpreter sukzessive abgearbeitet. Die Bewegungsanweisungen werden von einem Interpolator in kleine Einzelschritte

unterteilt und über eine Koordinatentransformation die dazugehörigen Achswinkel berechnet. Diese Achswinkel dienen als Sollvorgabe für den Lageregler, der über die in den Antrieben vorhandenen Resolver und den Servoregler die Positionen der einzelnen Achsen regelt. Durch die Steueranweisungen im Applikationsprogramm werden verschiedene Variablen gesetzt oder gelesen, die wiederum über die Aktionssteuerung und die Sensordatenverarbeitung mit den Ein-/Ausgängen des Bussystems verbunden sind. (Siciliano & Khatib 2008; Weber 2009; Dubbel 2007)

Programmierung

Ein wichtiges Merkmal des Industrieroboters ist die freie Programmierung. Sie ermöglicht es dem Anwender alle bahnbezogenen Aktionen, wie Bahnart, Geschwindigkeit, Schleifen etc., vorzugeben und am Ende ein lauffähiges Programm mit den entsprechenden Steueranweisungen (Setzen von Ein- und Ausgängen) zur Steuerung der Peripheriegeräte zu erstellen. Das Roboterprogramm, auch Applikationsprogramm oder Anwenderprogramm genannt, wird in der Regel in Funktionen gegliedert, die gekapselte Steueranweisungen enthalten. (Weck 2006; Neugebauer 1997)

Unter einem Programmierverfahren ist das planmäßige Vorgehen zur Erzeugung von Anwenderprogrammen zu verstehen. Nach VDI-Richtlinie 2863 ist ein Anwenderprogramm eine Sequenz von Anweisungen mit dem Zweck, eine vorgegebene Fertigungsaufgabe zu erfüllen. Die Vielzahl der zur Verfügung stehenden Programmierverfahren für Industrieroboter lassen sich nach Neugebauer (1997) und Siegert & Bocionek (1996) in direkte (auch prozessnahe oder Online-) und indirekte (auch prozessferne oder Offline-) Programmierverfahren unterteilen (vgl. Abbildung 5). Zu den direkten Programmierverfahren zählen die Teach-In- und die Play-back-Programmierung. Die indirekten Programmierverfahren beinhalten die textbasierte, die simulationsbasierte Programmierung sowie das Ablaufdiagramm und die aufgabenorientierte Roboterprogrammierung (Humburger 1998). Zudem gibt es verschiedensten Programmiermethoden, die derzeit in der Forschung und Entwicklung erprobt werden. Beispiele hierfür sind die aufgabenorientierte oder gestenbasierte Roboterprogrammierung (Vogl 2008; Lambrecht et al. 2011). Im industriellen Einsatz sind die Teach-In-Programmierung und die simulationsgestützte Offline-Programmierung die beiden dominierenden Programmiermethoden (Humburger 1998).

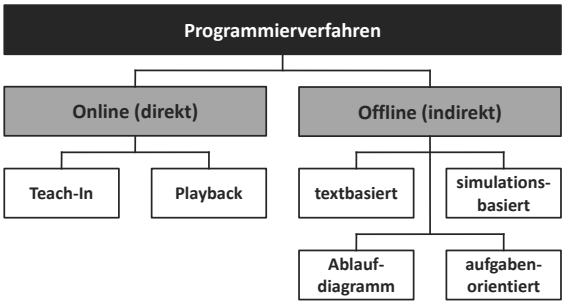

Abbildung 5: Übersicht über gängige Programmierverfahren für Industrieroboter

Online Teach-In-Programmierung – Der Bediener verfährt bei der Teach-In-Programmierung den Roboter mit dem Bediengerät (Programmierhandgerät) an eine spezifische Position und speichert diese ab (Weber 2009). Die Bewegungsparameter, wie Bewegungsart oder Geschwindigkeit, zwischen den gespeicherten Positionen gibt der Bediener explizit an (Weck 2006). Die Ansteuerung der Peripheriegeräte (z. B. Greifer) programmiert er entweder mit vorgefertigten Logik- und Steuerbefehlen oder er verwendet konfigurierbare Technologiepakete.

Simulationsgestützte Offline-Programmierung – Simulationssysteme zur Roboterprogrammierung basieren entweder auf erweiterten CAD-Systemen oder auf speziellen, teilweise herstellerspezifischen Lösungen mit 3D-Funktionalität. In diesen wird der Ablauf simuliert, das Roboterprogramm erstellt und das Programm anschließend unter Verwendung von Postprozessoren auf die Robotersteuerung übertragen. Zur Modellierung der Umgebung verfügen diese Systeme neben Bibliotheken mit Robotermodellen über Importschnittstellen für gängige 3D-Datenformate. (Weck 2006; Dubbel 2007)

2.3 Grundlagen Robotersysteme

2.3.1 Struktur von Robotersystemen

Ein Roboter als Manipulator ist ohne Peripherie nutzlos. Erst durch die Erweiterung mit Sensoren, Aktoren oder kombinierten, komplexen Geräten wird

der Roboter befähigt, seine dedizierte Aufgabe durchzuführen. Nach der Definition der DIN 8373 (1996) umfasst die Bezeichnung Robotersystem den Industrieroboter sowie Endeffektoren und Geräte bzw. Sensoren und Aktoren, die zur Ausführung der Aufgabe erforderlich sind. Darüber hinaus beinhaltet dieser Begriff alle Datenübertragungsschnittstellen, über welche der Roboter die Geräte, Sensoren und Aktoren überwachen bzw. steuern kann.

Die Ansteuerung der Peripheriegeräte erfolgt in Echtzeit über digitale oder analoge Ein- bzw. Ausgänge, die in der Robotersteuerung durch das Roboterprogramm gesetzt oder gelesen werden. Als Übertragungsmedium wird klassisch je Aus- bzw. Eingang eine Signalleitung verwendet, die sogenannte Punkt-zu-Punkt-Verdrahtung. In modernen Anlagen sind es meist Feldbusse, welche die Signale zu den Peripheriegeräten übermitteln (Langmann 2010).

In Abbildung 6 ist beispielhaft eine gängige Kommunikationsstruktur eines Robotersystems aufgezeigt. Der Roboter ist als Zellensteuerung für die Kommunikation und Steuerung sämtlicher Peripheriegeräte zuständig.

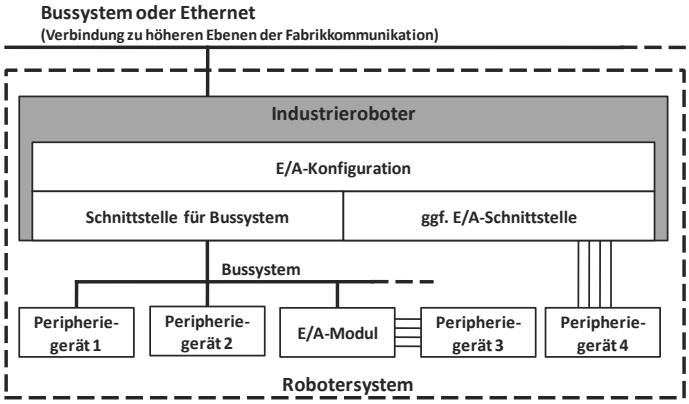

Abbildung 6: Schnittstellen eines Robotersystems in Anlehnung an Siciliano & Khatib *(2008)*

2.3.2 Peripheriegeräte und Endeffektoren

Als Peripheriegeräte werden diejenigen Einrichtungen bezeichnet, welche in Verbindung mit dem Roboter eingesetzt werden und zusammen das Roboter-

system ergeben. Zu diesen Geräten zählen beispielsweise Greifer, Fließbänder, Dreh- und Schwenkvorrichtungen oder Zuführeinrichtungen, die zusätzlich zu dem Roboter benötigt werden. (Nof 1999)

Peripheriegeräte werden allgemein in Sensoren, Aktoren und kombinierte Sensor-Aktor-Systeme unterteilt. Mit dem Fokus auf die Konfiguration ist eine Einordnung der Geräte hinsichtlich ihrer Signalverarbeitungsfähigkeit sinnvoll. In Anlehnung an Hesse & Schnell (2004) und Reinhart et al. (2011) erfolgt eine Einteilung in Basisgeräte, integrierte Geräte und sogenannte intelligente Geräte (vgl. Abbildung 7).

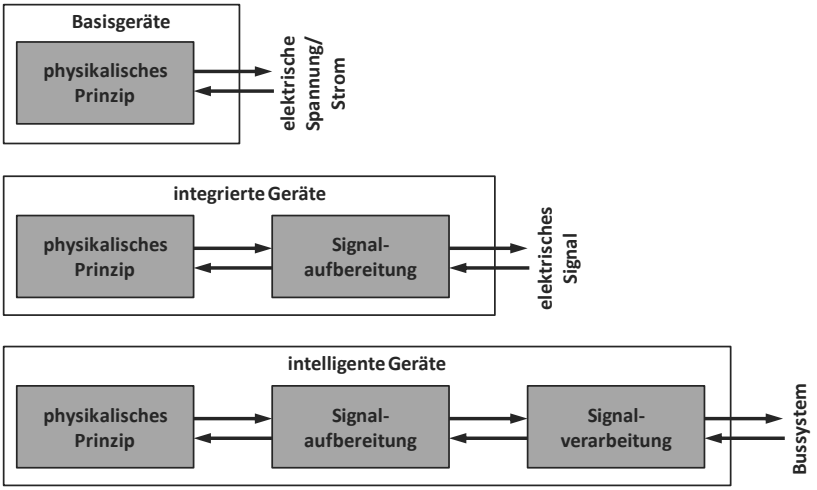

Abbildung 7: Einteilung der Peripheriegeräte in Anlehnung an Reinhart et al. *(2011)*

Basisgeräte wandeln physikalische Größen in elektrische Größen um bzw. umgekehrt. Ein Beispiel hierfür ist das Reedrelais, welches die Änderung des einwirkenden Magnetfelds in ein binäres Spannungssignal wandelt. Bei *integrierten Geräten* erfolgt zudem eine Verarbeitung, Aufbereitung oder Filterung der elektrischen Signale. Die *intelligenten Geräte* verfügen darüber hinaus über eine Anbindung an das Kommunikationsnetz des Robotersystems. Die Signale werden verarbeitet und über eine definierte Schnittstelle, z. B. ein Bussystem, weitergeleitet. Innerhalb des intelligenten Gerätes kann dabei selbst eine lo-

gische oder arithmetische Verknüpfung und Verarbeitung der Prozesssignale erfolgen. Das bedeutet, dass je nach Ausprägung diese Geräte parametrierbar, konfigurierbar und/oder programmierbar sind. Mit der Tendenz hin zu dezentralen Steuerungen und leistungsfähigeren Peripheriegeräten nimmt die Bedeutung der intelligenten Peripherie- bzw. Feldgeräte zu. (Fussel 2000)

Eine Spezialform der Peripheriegeräte stellen *Endeffektoren* dar. Sie werden als Vorrichtungen beschrieben, die dafür konzipiert sind, an der mechanischen Schnittstelle des Roboters befestigt zu werden und dem Roboter dazu dienen, seine Aufgabe zu erfüllen (DIN 8373 1996). Beispiele für Endeffektoren sind Greifer, spanende Roboterwerkzeuge, Prüfmittel, Sauger, Messzeuge, Schrauber, Farbspritzpistolen oder Punktschweißzangen (Hesse & Malisa 2010). Die Geometrie des Endeffektors und die Position des Arbeitspunktes zur mechanischen Schnittstelle haben durch die Veränderung der kinematischen Kette einen beträchtlichen Einfluss auf die Bewegungen des Roboters.

2.4 Industrielle Kommunikation

2.4.1 Architektur von Informationsnetzen

In vielen Bereichen des alltäglichen Lebens werden die unterschiedlichsten Kommunikations- und Informationsnetzwerke eingesetzt. Beispiele hierfür sind das Telefon- und Handynetz, das Local-Area-Network (LAN), z. B. für den Anschluss des PCs an das Firmennetz, Bluetooth, Bussysteme in modernen Kraftfahrzeugen oder auch Echtzeitbussysteme in industriellen Anlagen. Um diese teils komplexen Strukturen, Architekturen und Mechanismen von Informationsnetzen besser einordnen und verstehen zu können, wurde von der Internationalen Organisation für Normung (ISO) das OSI-Referenzmodell herausgegeben (ISO/IEC 7498 1994).

Netzwerke und kommunikationstechnischen Verbindungen lassen sich in dieses Referenzmodell einordnen, auch wenn sie nicht zwingend den exakten Konventionen folgen. Die Funktionen der Kommunikation sind in diesem Modell in sieben Schichten (englisch: Layer) unterteilt (vgl. Abbildung 8). Die Aufgaben der einzelnen Schichten werden von Kommunikationsprotokollen übernommen, die, je nach Netzwerk, unterschiedlich ausgeprägt und in verschiedenen (Software-)Systemen eingebettet sind. Die Transitionen zwischen den Schichten

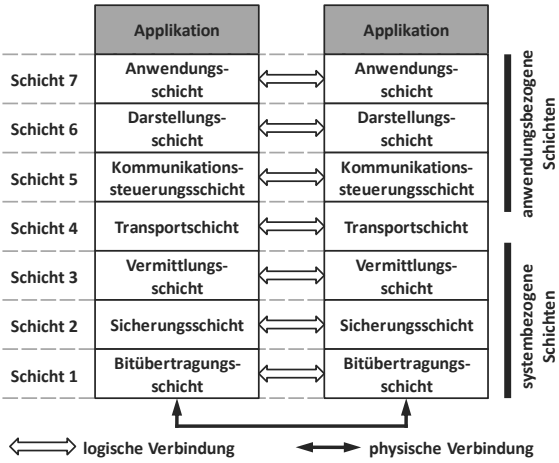

Abbildung 8: OSI-Referenzmodell nach Halsall *(1996)*

sind in dem jeweiligen Netzwerk vereinheitlicht und stellen für den nächsten Layer eine definierte Schnittstelle bereit. Der Verbund der für die Kommunikation erforderlichen Protokolle wird als Protokollstack bezeichnet. (Kauffels 1996)

Das OSI-Referenzmodell wird in transportorientierte und anwendungsorientierte Schichten unterteilt.

Transportorientierte Schichten – Die Schichten 1 bis 4, Bitübertragungsschicht, Sicherungsschicht, Vermittlungsschicht und Transportschicht sind meist inhärente Funktionen des Bus- bzw. Übertragungssystems und behandeln nachrichtentechnische Aspekte. Im Einzelnen definieren sie den physikalischen Übertragungsweg (Schicht 1), stellen die Datenübertragung sicher (Schicht 2), transportieren Datenpakete vom Sender zum Empfänger (Schicht 3) und kapseln die Nachrichtentechnik gegenüber den höheren Schichten der Systemtechnik ab (Schicht 4). (Kerner 1995)

Anwendungsorientierte Schichten – Prozesse, die für die Anwendungsseite relevant sind, werden von den Schichten 5 bis 7 bearbeitet. Die Kommunikationssteuerungsschicht (Schicht 5) regelt den Betriebsablauf der kommunizierenden Partner. Hier werden bspw. Verbindungen hergestellt und verwaltet, unterbrochene Übertragungen wieder aufgenommen und Synchronisierungs-

aufgaben durchgeführt. Diese Schicht ist vorwiegend im Betriebssystem der Kommunikationspartner angesiedelt. Die Darstellungsschicht (Schicht 6) stellt Ausdrucksmittel und Adressierungskonventionen zur Verfügung, die es z. B. ermöglichen, Datentypen zu definieren oder Formate festzulegen. Alle an der Kommunikation beteiligten Geräte müssen diesen Konventionen folgen. Die höchste Schicht des Referenzmodells ist die Anwendungsschicht (Schicht 7). Sie stellt die exklusive Schnittstelle für das Anwendungsprogramm dar, um einen Zugang zum Netzwerk zu erhalten. Sie ermöglicht diesem, meist über sogenannte Services, auf die Funktionen des Netzwerks zuzugreifen. (Kerner 1995)

2.4.2 Rahmenbedingungen industrieller Kommunikation

Moderne Produktionstechnik muss sich nahtlos in die Informationsverarbeitung der Fabrikumgebung einfügen können und somit die entsprechenden technischen sowie funktionellen Rahmenbedingungen erfüllen. In DIN 62264, Teil 3 (2007) wurde ein Ebenenmodell veröffentlicht, welches die verschiedenen Aufgaben der Kommunikation in einer Produktionsumgebung darstellt (vgl. Abbildung 9).

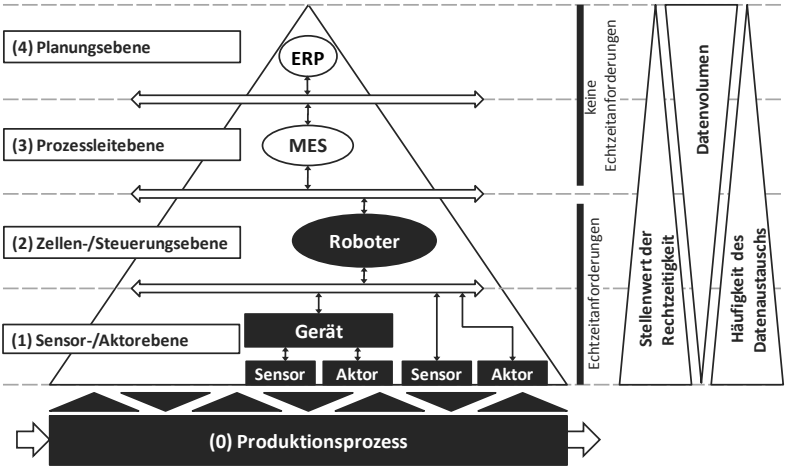

Abbildung 9: Automatisierungspyramide in Anlehnung an Milberg *(1992),* Schnell *(2006) und* DIN *62264, Teil 3 (2007)*

Die informationstechnische Verbindung zwischen einem Industrieroboter und den entsprechenden Peripheriegeräten – Sensoren, Aktoren und Geräten – findet sich in und zwischen den Ebenen 1 und 2 wieder. Neben gewissen Rahmenbedingungen, bspw. hinsichtlich elektromagnetischer Verträglichkeit oder Schutz gegen Vibration und Staub, stellen diese Ebenen folgende Anforderungen an die Kommunikation (Jäger 2009):

Topologie – Die physische Netztopologie sollte flexibel gestaltbar sein, um eine optimale Vernetzung innerhalb eines Systems zu gewährleisten (Jäger 2009). Gängige Topologie-Arten sind die Linien-, Baum-, Ring- und Sterntopologie (vgl. Abbildung 10) (Schnell 2006). Bei komplexeren Netzwerken sind oft auch Mischformen anzutreffen.

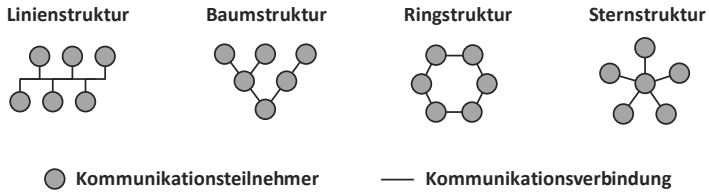

Abbildung 10: Gängige Topologie-Arten von Kommunikationsnetzen nach Schnell *(2006)*

Zeitverhalten – Je prozessnäher die Kommunikation ist, desto höher ist der Stellenwert der Rechtzeitigkeit und die Häufigkeit des Datenaustausches. Der Zeitrahmen bewegt sich in der Planungsebene im Bereich von Tagen bis Monate und in der Steuerungs- bzw. Sensor-Aktor-Ebene im Bereich von Millisekunden (DIN 62264 2007). Das Datenvolumen nimmt mit der Prozessnähe ab. Tabelle 1 zeigt typische Kommunikationsanforderungen hinsichtlich Verzögerungszeit und Datenvolumen auf den verschiedenen Ebenen der Automatisierungspyramide. Oft wird bei der zeitkritischen Kommunikation der Ebenen 1 und 2 auch von harten Echtzeitanforderungen gesprochen. Harte Echtzeit verlangt, dass auf bestimmte Ereignisse ohne Ausnahme innerhalb einer bestimmten Zeit geantwortet wird. Die Antwort muss dabei zwingend vorhersagbar sein und darf nicht von äußeren Umständen, wie bspw. anderen Systemaktivitäten, abhängen. Die Echtzeit erfordert nicht nur eine deterministische Übertragungsdauer (Latenz), sondern sie muss auch gleichermaßen taktsynchron mit vorhersagbaren Abweichungen erfolgen (Jitter). (Jäger 2009)

von \ zu	Extra- und Internet	Prozessleit- und Planungsebene	Steuerungsebene	Sensor-Aktor
Extra- und Internet	-	Verzögerung: 50 ms – 5 s Nutzdatenlänge: 1kB – 50 MB	-	-
Prozessleit- und Planungsebene	Verzögerung: 50 ms – 5 s Nutzdatenlänge: 1kB – 50 MB	Verzögerung: 5 – 50 ms Nutzdatenlänge: 1kB – 10 MB	Verzögerung: 5 – 50 ms Nutzdatenlänge: 1 – 500 kB	-
Steuerungsebene	-	Verzögerung: 5 – 50 ms Nutzdatenlänge: 1 – 500 kB	Verzögerung: 1 – 10 ms Nutzdatenlänge: 10 – 500 Byte	Verzögerung: 20 µs – 1 ms Nutzdatenlänge: 1 – 32 Byte
Sensor-Aktor	-	-	Verzögerung: 20 µs – 1 ms Nutzdatenlänge: 1 – 32 Byte	Verzögerung: 20 – 100 µs Nutzdatenlänge: 1 Bit – 10 Byte

Tabelle 1: Typische Kommunikationsanforderungen der verschiedenen Ebenen nach Furrer *(2000)*

Neben Jasperneite (2005) und IEC 61784, Teil 2 (2010) besteht eine Unterteilung in vier Echtzeitklassen der inzwischen aufgelösten IAONA (Industrial Automation Open Networking Alliance) (Lüder 2005), welche die Echtzeitfähigkeit hinsichtlich Latenz, Jitter und Bandbreite beurteilt. Abbildung 11 ordnet diese Klassen entsprechenden Anwendungsgebieten zu. Demnach ist für Industrieroboter Klasse 3 erforderlich.

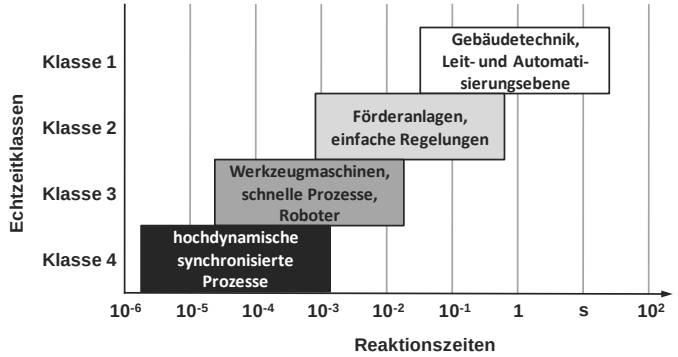

Abbildung 11: Echtzeitklassen nach Schaffeld *(2011)*

Zuverlässigkeit – Der Ausfall einer Produktionsanlage kann enorme Kosten verursachen (Jäger 2009). Die eingesetzten Kommunikationssysteme, die zu einem Stillstand führen können, müssen daher entsprechend robust und zuverlässig ausgeführt sein. Für den Fall einer Fehlübertragung sind bei industriellen Netzen meist Fehlererkennungs- und Korrekturverfahren zur Detektion und Behebung integriert. Bei einer solchen Störung im Netz ist eine Diagnosefähigkeit hilfreich, die eine schnelle Lokalisation und Störungsbeseitigung erlaubt (Furrer 2000).

Uhrzeitsynchronität – Zur Sicherstellung der Reihenfolge und synchronen Abfolge von Aktionen und Messungen in Produktionssystemen ist eine Synchronisierung der Kommunikationspartner erforderlich. Dies ist auch bei der Diagnose hilfreich, um Fehler von Folgefehlern zu unterscheiden. (Jäger 2009)

Wirtschaftlichkeit – Ein weiteres wichtiges Kriterium bei Kommunikationsnetzen ist die Wirtschaftlichkeit. Vor allem in den unteren Kommunikationsebenen – z. B. bei günstigen Sensoren oder Aktoren – sollten die Anbindung und die Infrastruktur 20 % der Gerätekosten nicht übersteigen. (Schnell 2006)

2.4.3 Industrielle Netzwerke

Den Kommunikationsanforderungen im industriellen Umfeld wird mit speziellen Netzwerken, sogenannten Bussystemen, begegnet. Während in den oberen Ebenen 3 und 4 der Automatisierungspyramide oft konventionelle Ethernet-Netzwerke Verwendung finden, werden in den unteren Ebenen 1 und 2 meist echtzeitfähige Feldbusse oder zunehmend auch Industrial-Ethernet-Netzwerke eingesetzt. Allgemein sind Bussysteme – im Gegensatz zur Zweipunktverdrahtung – dadurch gekennzeichnet, dass die Busteilnehmer über je eine Busschnittstelle und mit Ausnahme der sternförmigen Netztopologie auch gemeinsam über eine Datenleitung kommunizieren (vgl. Abbildung 12) (Schnell 2006; Furrer 2000).

Die verschiedenen, im Einsatz befindlichen Bussysteme lassen sich nach Schnell (2006) hinsichtlich ihres Zugriffverfahrens unterscheiden. Diese Verfahren sind notwendig, da für eine erfolgreiche Kommunikation über die gemeinsam genutzte Datenleitung nur ein Teilnehmer gleichzeitig senden darf.

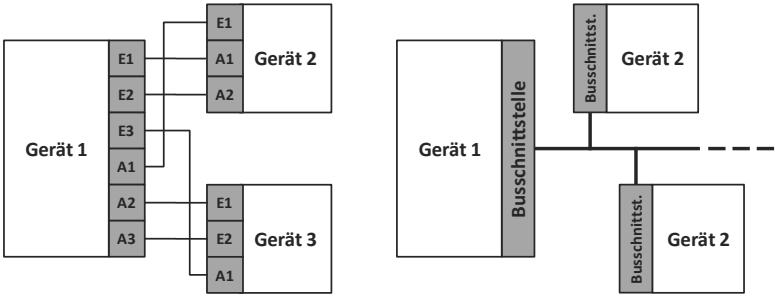

Abbildung 12: Gegenüberstellung Zweipunkt-Verdrahtung (links) und Bussystem (rechts)

Kontrollierte Buszugriffsverfahren – Bei der zentralen Buszuteilung, einem kontrollierten Buszugriffsverfahren, wird die Kommunikation durch einen Master geregelt, der über Datenanfragen Senderechte an die einzelnen Teilnehmer vergibt. Die dezentrale Buszuteilung erfolgt durch eine spezielle Nachricht (Token), die von Teilnehmer zu Teilnehmer weitergesendet wird. Der Empfänger (Tokenhalter) hat das Senderecht.

Zufällige Buszugriffsverfahren – CSMA steht für Carrier Sense Multiple Access. Will ein Teilnehmer eine Nachricht senden, hört er zunächst die Busleitung ab (Carrier Sense). Ist diese frei, wird die Nachricht gesendet; ist sie belegt, wird versucht, die Nachricht zu einem späteren Zeitpunkt erneut zu übertragen (Multiple Access). Senden zufälligerweise zwei Teilnehmer gleichzeitig eine Nachricht – diese Wahrscheinlichkeit steigt mit hoher Busauslastung – gibt es verschiedene Strategien, um die Übertragung zumindest eines Teilnehmers zu sichern (vgl. Abbildung 13): die Kollisionserkennung – CD (Collision Detection) und die Kollisionsvermeidung – CA (Collision Avoidance). Da Systeme mit zufälligen Buszugriffsverfahren (bspw. Industrial Ethernet CSMA/CD oder CAN-Netzwerke CSMA/CA) von Grund auf nicht echtzeitfähig sind, muss in diesen Bussystemen das Vergabeverfahren der Zugriffsrechte in höheren Schichten implementiert werden, um eine Echtzeitkommunikation ermöglichen. (Schnell 2006; Furrer 2000)

Letztendlich dient die Kommunikation der Übertragung von Prozessdaten, also Daten, die einen Bezug zum physikalischen Prozess haben. Neben der

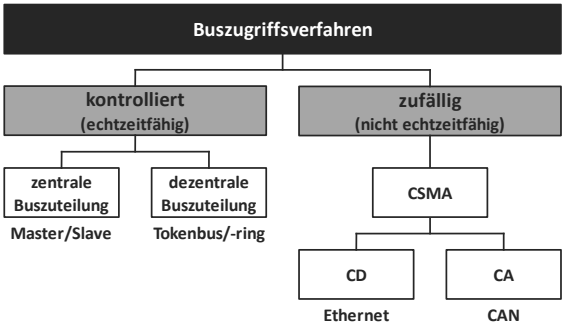

Abbildung 13: Übersicht über Buszugriffsverfahren nach Schnell *(2006)*

Client/Server-Architektur kommt zur Zuteilung der Prozessdaten zu einem Empfänger das Provider/Consumer-Modell zum Einsatz.

Die folgenden Abschnitte beschreiben die grundlegende Funktionsweise von Feldbussen und Industrial-Ethernet-Netzwerken, den Aufbau von Feldgeräten und das Zusammenspiel von Industrierobotern und Bussystemen.

Feldbusse

Konventionelle Feldbussysteme nutzen in der Regel nur die Schichten 1, 2 und 7 des ISO/OSI-Modells (vgl. Abbildung 14). Aufgrund der zeitlichen Anforderungen und der geringen Datenmengen ist die Reduktion des Funktionsumfangs auf diese drei Schichten erforderlich und auch möglich: Die Schichten 3 und 4 können entfallen, da zum einen die Adressierung der Teilnehmer meist direkt erfolgt und so unmittelbar mit der zweiten Schicht verbunden werden kann und zum anderen findet die Übertragung in Telegrammen statt, was eine Transportschicht überflüssig macht. Die Schichten 5 und 6 sind in der Regel durch die Applikation substituiert. Dies ist dadurch bedingt, dass der Kommunikationsaufbau über verschiedene Systemzustände und damit über die Schichten 5 und 6 erfolgt (z. B. preoperational und operational). Neben der reinen Kommunikation sind bei Feldbussen schichtenübergreifende Netzwerkmanagementfunktionalitäten essenziell, die bspw. Fehler in der Kommunikation melden oder mit denen die Busadresse einer Station eingestellt werden kann. (Fussel 2000)

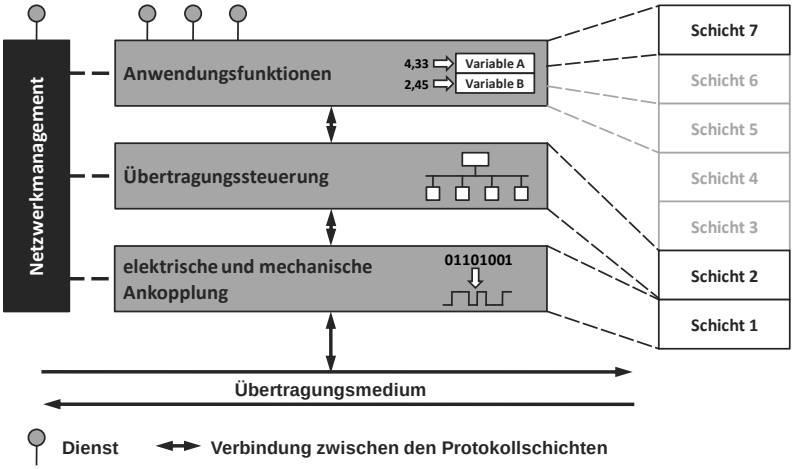

Abbildung 14: Gängige Schichtenstruktur von Feldbusprotokollen nach Fussel *(2000)*

Industrial-Ethernet-Netzwerke

Industrial-Ethernet-Netzwerke basieren auf der Ethernet-Technologie, die auch für die Heim- und Bürokommunikation eingesetzt wird. Mit dem Paradigmenwechsel von zentralistischen Systemen hin zu einer verteilten Automation und einer durchgängigen Kommunikation werden Ethernet-basierte Netzwerke immer mehr im Feldbereich eingesetzt (Schwager 2004a). So setzen laut einer Umfrage der Fachzeitschrift *Elektrische Automatisierung + Antriebstechnik* schon mehr als 70 % der Industrieanwender Ethernet-basierte Busse ein (Kuppinger 2004). Durch die Nutzung bestehender Teilsysteme und Produkttechnologien (bspw. Netzwerkhubs) sind deren Komponenten kostengünstig. Die verschiedenen Bussysteme unterscheiden sich teilweise nur durch den proprietären Protokollstack (Jäger 2009). Da bei Ethernet aufgrund von CSMA/CD Kollisionen auftreten und so nicht planbare Latenzzeiten entstehen, ist dieses Netzwerk an sich nicht echtzeitfähig (Schwager 2004b).

Anbieter von Industrial-Ethernet-Bussystemen haben verschiedene herstellerspezifische Prinzipien entwickelt, Echtzeitfähigkeit bei Ethernet-Netzwerken

zu ermöglichen. Diese sind in unterschiedlichen Layern der Kommunikation angesiedelt. (Jäger 2009)

Abbildung 15 zeigt verschiedene Telegrammstrukturen der gängigen Industrial-Ethernet-Bussysteme Sercos-III, Profinet, EtherCat und Powerlink.

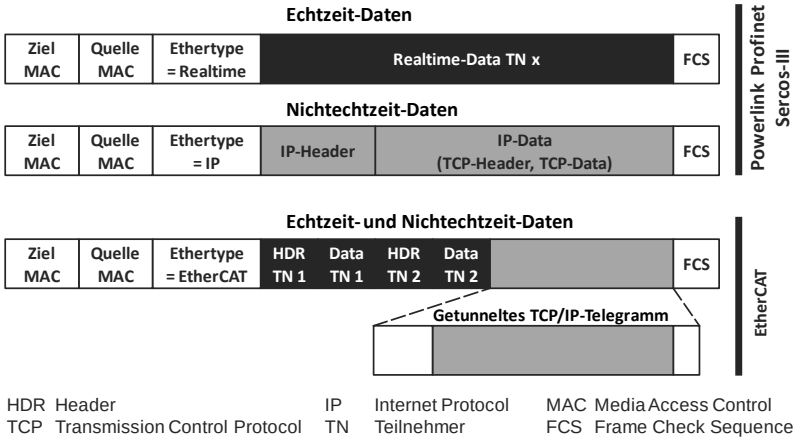

Abbildung 15: Telegramme für den Echtzeitbetrieb von Ethernet-basierten Feldbussen nach Jäger *(2009)*

Neben dem Prinzip zur Sicherstellung der Echtzeitfähigkeit unterscheiden sich die verschiedenen Industrial-Ethernet-Netzwerke vor allem durch ihren individuellen Protokollstack. Ein großer Vorteil aller Prinzipien der Industrial-Ethernet-Netzwerke ist, dass sie es ermöglichen neben den echtzeitkritischen Prozessdaten über gängige Netzwerkprotokolle, wie TCP oder UDP nicht zeitkritische Daten, wie Geräteparameter oder Maschinendaten zu übertragen. (Schwager 2004a, b)

Abbildung 16 zeigt beispielhaft den Aufbau des Industrial Ethernet Protokollstacks von Powerlink. Die echtzeitkritischen Prozessdaten werden direkt über den *Powerlink Data Link Layer* in das Prozessabbild (PDO) die Powerlink Anwendungsschicht geschrieben. Nicht zeitkritische Daten können über gängige Ethernet-Protokolle und Dienste, wie TCP/IP, UDP/IP und FTP bzw. HTTP interpretiert und übertragen werden. Begleitet wird der Protokollstack von einem schichtenübergreifenden Netzwerkmanagement.

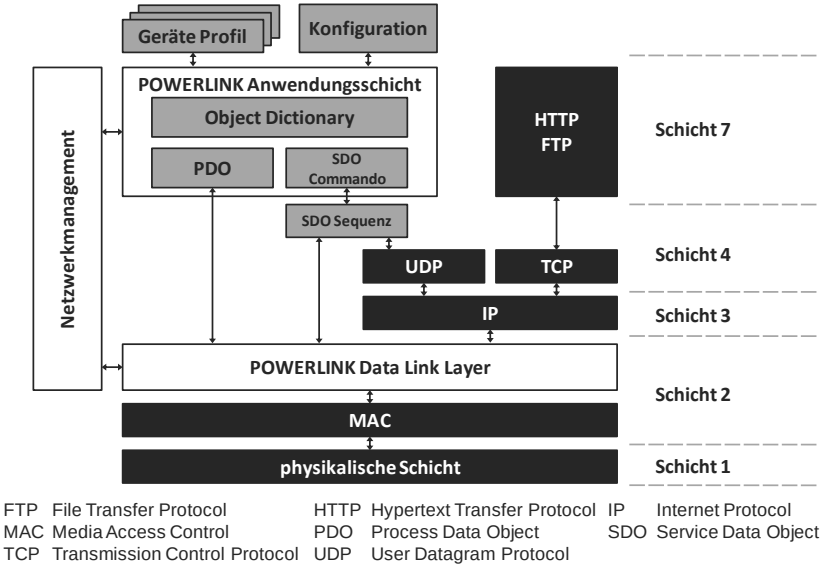

*Abbildung 16: Protokollstack von Industrial-Ethernet-Netzwerken am Beispiel von Powerlink (*EPSG *2008)*

2.5 Konfiguration und Inbetriebnahme von Robotersystemen

Von der Planung des Robotereinsatzes bis hin zum (erneuten) Produktionsbetrieb eines robotergestützten Produktionssystems sind – wie bereits erwähnt – diverse Schritte erforderlich. Zur Strukturierung dieser Aufgaben wurde der Gesamtprozess analysiert und unter Berücksichtigung bestehender Erkenntnisse in drei Phasen gegliedert (vgl. Abbildung 17): Planungsphase, Systemintegration und Betriebsphase.

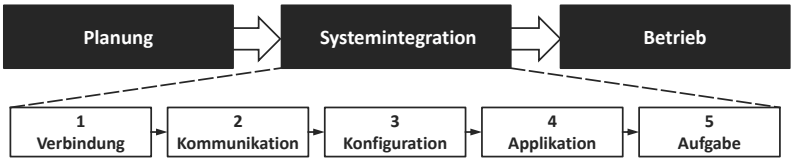

Abbildung 17: Ablaufdiagramm für den Aufbau und Konfiguration eines Robotersystems

Zur besseren Darstellung der theoretischen Beschreibung des Integrationsablaufs werden die verschiedenen Phasen durch ein Beispiel einer „Pick-and-Place-Anwendung“ ergänzt.
Beispiel: Ein Industrieroboter, der mit einem Bussystem ausgestattet ist, um Peripheriegeräte anzubinden, soll um einen einfachen Greifer erweitert werden.

In der *Planungsphase* sind alle erforderlichen Schritte zusammengefasst, die vor dem Hardwareaufbau durchgeführt werden. Das beinhaltet z. B. die Hardwareauswahl oder das Layout der Roboterzelle. Abhängig von der angestrebten Systemperformance sind zusätzliche Arbeiten, wie die Ermittlung und Planung der Zykluszeit, erforderlich. Darüber hinaus werden üblicherweise alle Vorgänge der folgenden Systemintegration vorbereitet. Aufgrund der hohen Komplexität der Betriebsmittel und des Integrationsaufwands ist eine detaillierte Planungsphase essenziell.
Beispiel: Es wurde ein pneumatischer Greifer mit einem Ventil ausgewählt, der durch ein digitales Signal geöffnet und geschlossen werden kann. Weiter sollen zwei Sensoren den Zustand des Greifers erkennen. Der Greifer soll mit einer Busschnittstelle ausgestattet werden. Seine Befestigung am Roboterflansch wird bestimmt und ein Anschlussschaltplan entworfen.

Die *Systemintegration* – der Hardwareaufbau und die Konfiguration – ist hier in Anlehnung an Siciliano & Khatib (2008) in fünf Schritte unterteilt. Diese werden typischerweise in der dargestellten Reihenfolge ausgeführt.

1. *Physische Verbindung* – Dieser Schritt ist von der Herstellung mechanischer Verbindungen dominiert. Dies beinhaltet u. a. das Aufstellen und Befestigen der Geräte, die Montage von Adapterplatten, den Anschluss von Pneumatik, Elektrik oder Netzwerkkabeln (z. B. Feldbusverbindung).
   Beispiel: Der Greifer wird an den Flansch des Roboters geschraubt und die Pneumatik-, die Signalleitungen sowie die Stromversorgung werden angeschlossen.

2. *Herstellung der Kommunikation* – Die logische Verbindung von einzel-Bit-digitalen oder multi-Bit-analogen E/A werden hergestellt. Diese Daten sind Rohdaten ohne jede Metainformation, sogenannte Prozessdaten. Der bit- oder byteweise Datentransfer über das Kommunikationsmedium, z. B. das Feldbussystem, wird definiert. Das erfordert die Einrichtung und

Parametrisierung des Netzwerks und der Kommunikationspartner.
Beispiel: Die Robotersteuerung und das Bussystem sind konfiguriert. Zu diesem Zweck wurde der Greifer in den Buseinstellungen als neues Feldgerät definiert. Zwei Eingangs-Bits und ein Ausgangs-Bit wurden in den Konfigurationseinstellungen hinterlegt. Die Bedeutung sowie das Format dieser Daten, die beiden Sensorsignale, welche den Zustand des Greifers angeben und der digitale Ausgangswert zum Öffnen und Schließen des Greifers sind zu diesem Zeitpunkt nicht bekannt. Zusätzliche Informationen, wie die Zykluszeit oder die Knotennummern der Geräte, werden ergänzt.

3. *Einstellung der Konfiguration* – Die Nachrichten zwischen den interagierenden Geräten werden konfiguriert, die Dienste des Kommunikationsmediums eingerichtet. Darüber hinaus wird die Interpretation der Prozessdaten hinsichtlich des Datenformats und der Datentypen durchgeführt. Andere Aspekte dieser Phase sind Feineinstellungen zur Verbesserung der Leistung und der Ressourcenauslastung.
Beispiel: Die Prozessdaten des Bussystems (die zwei Eingangs-Bits und das Ausgangs-Bit) werden logisch mit den Systemvariablen des Roboters verbunden. Diesen Variablen werden dafür üblicherweise Namen zugewiesen. Für manche Anwendungen werden in einigen Robotersteuerungen konfigurierbare Softwaremodule (z. B. KUKA GRIP-TECH) angeboten, die es ermöglichen, die Prozessdaten während der Roboterprogrammierung zu verändern. Zudem werden der Tool-Center-Point und andere vom Benutzer definierbare Koordinatensysteme festgelegt.

4. *Herstellung der Applikationsschnittstelle* – In dieser Phase werden Funktionen für die Anwendung geschrieben. Diese Funktionen definieren die individuelle Interaktion des Gerätes mit der Robotersteuerung über die vorher festgelegten Prozessvariablen. Die Inhalte dieser Roboterfunktionen werden exakt auf die Funktionalitäten der Geräte abgestimmt.
Beispiel: Im Falle der Pick-and-Place-Anwendung werden zwei Anwendungsfunktionen definiert: „öffnen“ und „schließen“. In jeder dieser Funktionen werden zur Verifizierung des Greiferstatus die jeweiligen Eingänge abgefragt. Der Ausgang wird gesetzt, um den Greifer zu öffnen und zu schließen. Die erfolgreiche Durchführung wird durch die Überwachung der Eingangsvariablen überprüft.

5. *Programmierung der Aufgabe* – Schließlich werden die Erstellungen des Roboterprogramms unter Zuhilfenahme der Anwendungsfunktionen vorgenommen. Dieser Schritt kann mit verschiedenen Programmierverfahren, wie Online- oder Offlineprogrammierung erfolgen. Die Anwendungsfunktionen werden innerhalb des Programmcodes aufgerufen.
   Beispiel: Mithilfe der Teach-In-Programmierung wird die Roboterbahn für die Pick-and-Place-Anwendung definiert. Die Funktionen „öffnen“ und „schließen“ werden im Programmcode aufgerufen. Während der Programmierung und nach der Fertigstellung wird das Programm getestet.

Im *Betrieb* ist das Robotersystem produktiv und führt die vorgesehenen Aufgaben aus. Alle Vorbereitungen sind abgeschlossen. Das Robotersystem wird in der Regel vorher getestet und läuft anschließend im Automatikbetrieb. Lediglich für Wartungsarbeiten oder im Störungsfall wird das System angehalten.
Beispiel: Das Programm für die Handhabungsaufgabe wurde bei voller Geschwindigkeit getestet und in den Automatikbetrieb überführt. Der Roboter wiederholt die Pick-and-Place-Aufgabe.

Dieses Beispiel zeigt die Komplexität und das Ausmaß des Konfigurationsaufwands für ein Robotersystem nicht nur bei der erstmaligen Inbetriebnahme, sondern auch für den Fall, dass Änderungen an dem System durchgeführt werden müssen, der Rekonfiguration. Bei größeren und umfangreicheren Systemen mit höher entwickelten Komponenten steigen der Umfang und die Vielschichtigkeit dieser Arbeiten stark an.

3 Stand der Technik und Forschung

3.1 Allgemeines

Die effiziente Weiterverwendung und Restrukturierung von Komponenten mit informationstechnischem Bezug sind Herausforderungen, die sich in vielen Bereichen technischer Systeme wiederfinden. Im Folgenden werden unterschiedliche bestehende Ansätze aufgezeigt und deren Einsatzfähigkeit für Plug&Produce-Robotersysteme diskutiert.

3.2 Automatische Konfiguration in Computernetzwerken

3.2.1 Allgemeines

Im Gegensatz zur Kommunikation in industriellen Netzwerken – hier müssen die Verbindungen oft noch bis in die unteren Kommunikationsschichten manuell konfiguriert werden – haben sich bei Office- und Heimcomputern selbstkonfigurierende Plug&Play-Netzwerke bereits weitestgehend etabliert. Durch den breiten Einsatz von Computertechnologie wurde es erforderlich, dass auch Laien Netzwerke konfigurieren sowie neue Geräte einbinden können. Es ist kein Experte, in manchen Fällen, bis auf das Anschließen, sogar überhaupt kein manueller Eingriff erforderlich, um bspw. einen neuen Drucker anzuschließen. Je nach Anforderungen der Applikation und der verwendeten Betriebssysteme haben sich auf Basis standardisierter Bus- und Netzwerksysteme unterschiedliche Plug&Play-Prinzipien etabliert. Die Reichweite der Umsetzung dieser Prinzipien richtet sich nach dem verwendeten Bus- und Betriebssystem und reicht von einfacher Adresszuweisung (z. B. Dynamic Host Configuration Protocol – DHCP bei Ethernet-Netzwerken) bis zur nahezu vollständigen Konfiguration (z. B. Plug&Play bei USB). Plug&Play ist immer ein Zusammenspiel zwischen Bus- bzw. Kommunikationssystem und Betriebssystem.

Im Folgenden werden die Prinzipien zweier bekannter Vertreter selbstkonfigurierender Netzwerke beschrieben: Enumeration am Beispiel von USB (Universal Serial Bus) und Common Base Protocols am Beispiel von UPnP™ (Universal Plug and Play).

3.2.2 Treiber-basierte Enumeration (USB)

Die Treiber-basierte Enumeration beginnt mit dem Prozess der Identifikation eines neu angeschlossenen Gerätes und dessen Eigenschaften. Auf dieser Informationsgrundlage wird anschließend der „passendste“ dem Betriebssystem zur Verfügung stehende Gerätetreiber ausgewählt. Vereinfacht dargestellt besteht die Enumeration aus fünf Schritten: (1) Anschluss und Auffinden neu angeschlossener Geräte, (2) Identifizierung der neuen Geräte, (3) Zuweisung einer eindeutigen Adresse, (4) Bestimmung der Gerätekonfigurationen und Funktionen, (5) Auswahl eines Gerätetreibers – dieser Vorgang ist abhängig vom eingesetzten Betriebssystem. Die Enumeration wird vor allem in Windows-unterstützten Netzwerken angewendet, da Schritt 5 mit der Windows-Gerätetreiber-Architektur kompatibel ist. (Axelson 2009; Hulzebosch 2008)

Im Falle von USB ist das Auffinden neu angeschlossener Geräte (Schritt 1) durch eine Spannungsüberwachung einer definierten Busleitung realisiert. Die weiteren Konfigurationsschritte (Schritte 2-4) erfolgen durch sukzessives Auslesen einer auf dem Gerät hinterlegten Informationsdatei (INF-Datei), welche eine Beschreibung der Geräteeigenschaften enthält. Schließlich wählt das Betriebssystem den für die Beschreibung besten, verfügbaren Treiber (Schritt 5) aus, der ggf. erst mit einer Treiber-CD installiert werden muss. (Axelson 2009; Hulzebosch 2008)
USB besitzt an dieser Stelle noch eine Besonderheit: In der USB-Spezifikation sind verschiedene Geräteklassen definiert. Die Einordnung in die Geräteklasse ist in der INF-Datei angegeben. Benötigt ein Gerät nur die Schnittstellen einer der vorgegebenen Geräteklasse, ermöglicht dies die Verwendung des Gerätes mit einem Standardtreiber in vollem Umfang. (Axelson 2009; Hulzebosch 2008)

3.2.3 „Common Base Protocols“ (UPnP™)

Im Gegensatz zur Enumeration kommt dieses Prinzip vollständig ohne Treiber aus. Die Konfiguration wird auf Basis eines umfangreichen Protokollstacks durchgeführt, der für die verschiedenen Schichten spezifische Protokolle bereitstellt. Abbildung 18 zeigt beispielhaft den Protokollstack für IP-basierte Netzwerke.

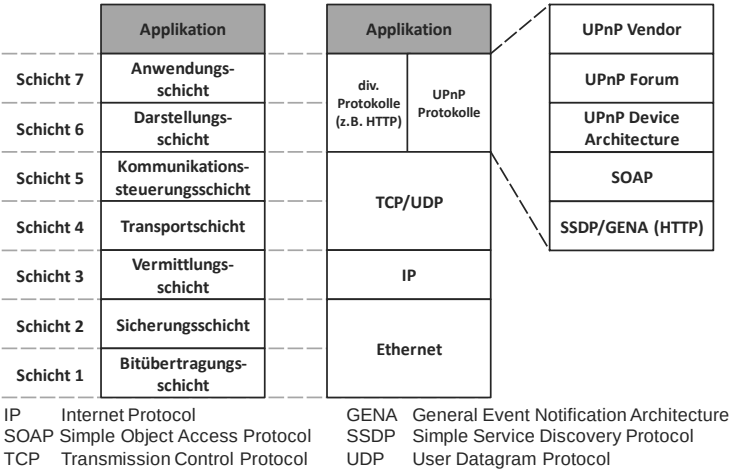

Abbildung 18: UPnP™-Protokollstack für IP-basierte Netzwerke nach UPnP (2008) *und Einordnung in das OSI-Referenzmodell nach* Stallings (2004)

Bei UPnP™ erfolgt die Konfiguration in verschiedenen meist zeitlich aufeinanderfolgenden Schritten. Die jeweils maßgeblichen Protokolle sind in Klammern aufgeführt:
(1) *Addressing* – Zuweisung oder Bestimmung einer eindeutigen IP-Adresse bspw. durch DHCP-Server (IP, UDP bzw. TCP und HTTP),
(2) *Discovery* – Bereitstellung von Basisinformationen über das Gerät, sodass sie im Netzwerk gefunden werden können (SSDP),
(3) *Description* – Übermittlung detaillierter Geräteinformationen und Dienste (UPnP Device Architecture),
(4-5) *Control und Eventing* – Steuerung der Geräte bzw. Benachrichtigung bei bestimmten Ereignissen an den Geräten (SOAP, GENA und UPnP Forum),
(6) *Presentation* (optional) – Möglichkeit der Geräteüberwachung über einen Browser (HTTP und UPnP Forum).

Die Steuerung der Geräte erfolgt hier über das Versenden von SOAP bzw. GENA Paketen im XML-Format über Standardprotokolle. Auch hier wird die Plug&Play-Funktionalität vom Betriebssystem bzw. von der Anwendung durch das Versenden bzw. Empfangen der Protokolle ermöglicht. (UPnP 2008)

Neben den beschriebenen Plug&Play-Netzwerken gibt es noch einige weitere, die nach oder ähnlich einem der beiden oben beschriebenen Prinzipien funktionieren.

3.3 Vereinfachte Konfiguration im produktionstechnischen Umfeld

3.3.1 Allgemeines

Während im Bereich der Heim- und Büro-Computer die automatische Einbindung von Peripheriekomponenten weit fortgeschritten ist, stehen derartige Mechanismen im produktionstechnischen Umfeld noch am Anfang. Grund dafür sind die bereits angesprochenen proprietären Geräte, Schnittstellen und Betriebssysteme sowie deren heterogener Funktionsumfang. Dies führt zu einer hohen Diversität an Komponenten.

Bemühungen, die Konfiguration für Informationsschnittstellen zu vereinfachen, erstrecken sich über eine weite Bandbreite der Produktionstechnik. Bestehende Erkenntnisse, Methoden und Vorgehensweisen sind hier anhand der maßgeblichen Funktionsprinzipien kategorisiert: Standardisierte Protokolle, Beschreibungsformen und modellbasierte Ansätze, Middleware und alternative Systemarchitekturen. Die Arbeiten werden hinsichtlich ihres Potenzials zur Konfiguration der Kommunikation, der Dateninterpretation und der Applikation im produktionstechnischen Umfeld untersucht. Des Weiteren wird die Anwendbarkeit des Standes der Technik auf die automatische Konfiguration von Robotersystemen analysiert.

3.3.2 Standardisierte Protokolle

Der Datenaustausch im produktionstechnischen Umfeld stellt vor allem aufgrund unterschiedlicher Softwarearchitekturen eine Herausforderung dar (Wellenreuther & Zastrow 2009). Im industriellen Einsatz haben sich Standards etabliert, die Softwarekomponenten, welche die Einbindung von Steuerungen in die Prozessleitebene ermöglichen, wiederverwendbar machen. Im Forschungsumfeld existieren darüber hinaus Bemühungen, für spezielle

Anwendungen Schnittstellen von Robotern und Peripheriegeräten zu vereinheitlichen. Die Formalisierung und Standardisierung des Datenaustausches bieten den Vorteil, dass keine schnittstellenspezifischen Anpassungen erforderlich sind. Es genügt die einmalige Implementierung.

OPC (OLE for Process Control) bietet beispielsweise eine derartige standardisierte Schnittstelle zwischen Automatisierungstechnik und Prozessleittechnik. So kann mit einer einzigen Schnittstelle auf die Prozesssteuerungen verschiedener Hersteller zugegriffen werden. Voraussetzung ist, dass Hersteller der Steuerungsgeräte diese befähigen, im OPC-Standard zu kommunizieren. Im Anwendungsprogramm der Steuerung ist explizit ein OPC-Client zu programmieren und zu konfigurieren, der mit dem OPC-Server der PC-basierten Prozessleitebene kommuniziert. Darüber hinaus ist eine unterlagerte, nicht standardisierte Kommunikationsverbindung, die nicht notwendigerweise echtzeitfähig ist (z. B. Ethernet-TCP/IP oder Profibus), zwischen OPC-Client und OPC-Server einzurichten. (Wellenreuther & Zastrow 2009)
Betrachtet man die Einordnung in das OSI-Referenzmodell, so standardisiert OPC die Datenformate in der Darstellungsschicht (Schicht 6). Sowohl die Kommunikationsschichten darunter als auch das Anwendungsprogramm darüber bleiben davon unberührt und bedürfen einer expliziten manuellen Konfiguration.

Der *Weihenstephaner Standard* – eine branchenspezifische Lösung im Bereich der Getränkeabfüllung – verfolgt eine weitergehende Standardisierung zwischen Steuerungs- und Prozessleitebene. Neben der Darstellungsebene werden die unteren Kommunikationsschichten, die physikalische Schnittstelle, das Netzwerk und die darin verwendeten Protokolle, ebenfalls standardisiert. Auch auf der Anwendungsebene sind die Daten und Funktionen, welche von Maschinen und Kontrollgeräten bereitgestellt werden, vereinheitlicht. Es sind Zustandsmaschinen für die verschiedenen Anlagentypen definiert, in der die möglichen Systemzustände einer Maschine und deren Transitionen vorgegeben sind. (Voigt & Kather 2005; Kather & Voigt 2008)
Ähnlich der USB-Schnittstelle wird eine Gerätebeschreibung übermittelt, über welche die angeschlossenen Maschinen identifiziert werden. Dies vereinfacht die Einbindung der Geräte in die Leitebene. Damit stellt die Standardisierung ein probates Mittel für Anlagen zur Getränkeabfüllung dar, den Engineeringaufwand zu reduzieren. Durch die weitgehende Standardisierung ist die Flexibilität

der Einbindung abweichender Anlagenkonzepte oder Funktionsumfänge mit einer Erweiterung bzw. Veränderung des Standards verbunden.

Der Austausch von Informationen setzt voraus, dass beide Kommunikationspartner in der gleichen Syntax und Semantik kommunizieren. Datenaustauschformate können hier den Engineeringaufwand reduzieren, da die Anzahl der zu entwickelnden Schnittstellen durch den Standard reduziert wird. In der CAD-Computertechnik werden solche Austauschformate bereits eingesetzt (z. B. IGES, VRML oder STEP).

Sauer (2008b) überträgt dieses Prinzip auf die Leitsystem-Projektierung. Dabei werden die verschiedenen, bereits vorhandenen Planungsdaten (ggf. auch von Zulieferern), welche in unterschiedlichen Formaten vorliegen können, in ein standardisiertes Austauschformat übersetzt und für das Leitsystem aufbereitet (vgl. Abbildung 19). (Sauer 2008a, b; Bär et al. 2008)

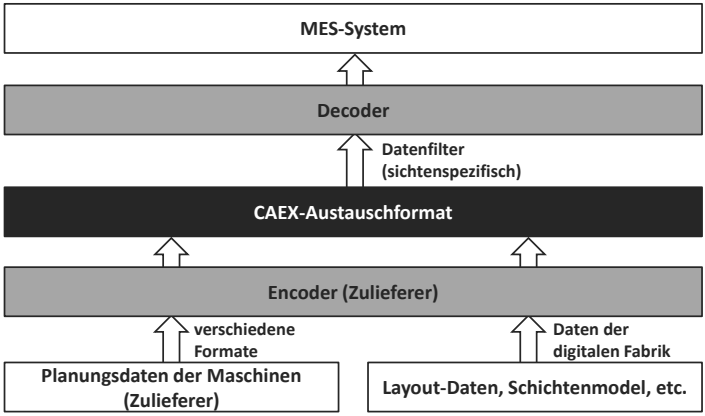

Abbildung 19: Datenaustausch für die Leitsystemprojektierung nach Bär et al. *(2008)*

Die Equipment-beschreibenden Informationen können, nachdem sie mit einem Konverter (Encoder) in das Austauschformat gebracht wurden, direkt vom Anlagenlieferanten an das ausführende Unternehmen übermittelt werden. Als Austauschformat wird hier CAEX (Computer Aided Engineering Exchange), ein XML-basiertes Datenformat zur Speicherung hierarchischer Objektinformationen, verwendet. Über einen weiteren Konverter (Decoder) werden ausgewählte

Informationen direkt dem MES-System zur Verfügung gestellt. Damit ist es möglich, MES-Systeme durch den Einsatz von in der Planungsphase erzeugten Daten zu parametrisieren.

ORiN (Open Resource Interface for the Network), ein Projekt der *Japan Robot Association*, ist bestrebt durch Standardisierung eine hersteller- und systemunabhängige Schnittstelle zwischen den Steuerungen der Produktionsressourcen und den PCs der Prozessleitebene oder anderen Benutzerschnittstellen, z. B. zur Fehleranalyse, zu etablieren. ORiN beschränkt sich auf die systemübergreifende, nicht-echtzeitfähige Kommunikation. Diese Zielsetzung soll durch die Bereitstellung standardisierter Funktionen erreicht werden, welche den Zugriff auf die Geräte unterschiedlicher Hersteller ermöglichen. Dabei wird auf bestehende De-facto-Standards zurückgegriffen, die für den Einsatz in ORiN angepasst wurden. (Mizukawa et al. 2002, 2004)
ORiN stellt Schnittstellen sowohl für Anbindung von Geräten als auch für die Anbindung verschiedener Applikationen bereit. So kann die Anzahl der Schnittstellen von einer n x m Beziehung auf eine n + m Beziehung reduziert werden (vgl. Abbildung 20). (JRA 2008)

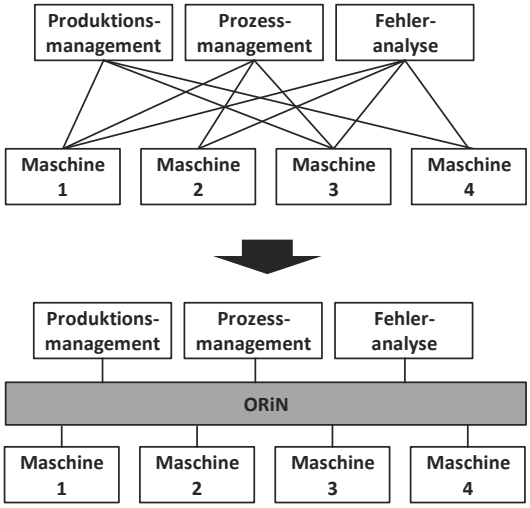

Abbildung 20: Prinzip von ORiN nach JRA *(2008)*

Neben den standardisierten Schnittstellen zur Anbindung der Automatisierungstechnik gibt es Bestrebungen, Standards für spezielle Anwendungen für Industrieroboter zu etablieren.

XIRP (XML Interface for Robots and Peripherals) ist ein XML-basiertes Protokoll, welches durch Standardisierung eines durchgängigen Kommunikationsprofils den Aufwand zur Integration von komplexen Sensoren reduzieren soll (Gauss et al. 2005). Den Ursprung hat diese Schnittstelle in dem vom BMBF geförderten Projekt ARIKT (Automatische roboterbasierte Inspektion komplexer Teile), welches am Institut für Prozesstechnik, Automation und Robotik des Karlsruher Instituts für Technologie durchgeführt wurde.
Dem im VDMA-Einheitsblatt 66430-1 (VDMA 66430) standardisierten Protokoll liegt eine bidirektionale Client-Server-Architektur zugrunde (Gauss et al. 2006). XIRP legt für die Maschine-Maschine-Schnittstelle den vollständigen Protokollstack fest (Sagert 2006). Die Steuerung lädt zur Identifikation des angeschlossenen Geräts eine standardisierte Gerätebeschreibungsdatei herunter, ähnlich wie USB (Schwarzkopf 2006). Durch die Verwendung des konventionellen TCP/IP Protokolls in XIRP, der erforderlichen Dateninterpretation des übertragenen XML-Protokolls beim Empfänger, kombiniert mit der Netzwerkarchitektur, kann nach VDMA (66430) eine zeitnahe Synchronisation der Daten erfolgen. Eine Übertragung in Echtzeit, wie sie bei konventionellen Übertragungssystemen in der Automatisierungstechnik üblich ist, kann, vor allem bei mehreren Teilnehmern im Netzwerk, aufgrund der verwendeten Protokolle nicht gewährleistet werden. Für Geräte ohne Echtzeitanforderungen, deren vollständiger Funktionsumfang im Standard integriert ist und welche den XIRP-Protokollstack implementiert haben, stellt dieser Standard eine signifikante Erleichterung bei der Geräteintegration dar.

Die Standardisierung von Schnittstellen ist eine bewährte Methode zur Steigerung der Interoperabilität. Sie trägt unter fixierten Rahmenbedingungen, wie bspw. Funktionsumfang oder Systemstruktur, zur erheblichen Erleichterung der Anbindung neuer Geräte bei. Jedoch ist eine Implementierung desselben Standards in den Systemen beider Kommunikationspartner erforderlich. Um eine Unterstützung der Konfiguration zu ermöglichen, ist in den jeweiligen Systemen eine entsprechende Funktionalität zu hinterlegen. Mit einem steigenden Grad an Ausdrucksstärke sinkt die Anwendungsbreite eines Standards (Epple 2003). Erweiterungen oder Veränderungen der Rahmenbedingungen erfordern

daher meist eine Anpassung des Standards, was neben dem Aufwand unter Umständen zu Problemen bei der Versionskompatibilität führen kann. Die Interoperabilität von (Quasi-)Standards kann durch Datenaustauschformate erhöht werden.

3.3.3 Beschreibungsformen und modellbasierte Ansätze

Gerätebeschreibungen, wie im Weihenstephaner Standard bereits ansatzweise vorhanden, bieten die Möglichkeit, Informationen über Geräte zur Verfügung zu stellen und für Konfigurationszwecke zu nutzen. Im Vergleich zur Standardisierung abstrahieren Beschreibungsformen sowie modellbasierte Ansätze die Schnittstellen und erlauben damit eine flexiblere gerätespezifische Festlegung einzelner Parameter.
Im Bereich der Feldbusse werden bereits heute Gerätebeschreibungen eingesetzt (z. B. EDDL), um die Einbindung der Geräte zu vereinfachen. Dafür werden diese Beschreibungen meist in ein Programmierwerkzeug geladen, interpretiert und die Informationen logisch miteinander verknüpft. Gerätebeschreibungen im Bereich der Feldbusse decken die unteren Kommunikationsschichten auf Bit- und Byte-Ebene ab. Neben dem Einsatz in der Feldbustechnologie finden diese Ansätze auch in Forschungsarbeiten Anwendung.

Das Forschungsprojekt *EmsA* (Erstellung eines Entwicklungssystems für modulare, selbstkonfigurierende Visualisierungen zur Anlagenüberwachung) hatte zum Ziel, gerätespezifische Visualisierungen zur Steuerung und Überwachung zu einem Gesamtvisualisierungssystem zu integrieren. Dazu wurden bestehende Beschreibungsformen um Visualisierungsdaten erweitert. Die sogenannte EEDD (Erweiterte Gerätebeschreibung) enthält Informationen über die Gerätesignale, über die Gerätevisualisierungen sowie Kopplungsinformationen, welche die Gerätesignale an die Visualisierungselemente bindet. Diese Informationen werden von den Geräten in einem Netzwerk bereitgestellt oder von einer Datenbank geladen und für ein Zielsystem zu einer Visualisierung zusammengestellt. Dafür sind folgende Teilaspekte von Bedeutung: automatische Registrierung von Änderungen an der Anlagentopologie, Identifikation der registrierten Komponenten, Integration der gefundenen Geräte in das Gesamtsystem, Erzeugung zusätzlicher Gerätesignale zu Überwachungs- und Diagnosezwecken. Die Prozessdaten, welche die Visualisierung anzeigen soll, werden abschließend über eine Bindungsdatei mit der Visualisierung verknüpft.

Da konventionelle Ethernet-Netzwerke für die Visualisierung ausreichende Ressourcen zur Verfügung stellen, setzt EmsA auf Standard-Ethernet-Protokolle und -Mechanismen, wie FTP oder DHCP. (Brecher et al. 2008; Jensen et al. 2009)

Ein gänzlich anderer Ansatz, Beschreibungsformen für die Konfiguration zu nutzen, wurde im Forschungsprojekt *Produflexil* (Flexible Anbindung von Produktionsanlagenmodulen durch Adaptivität und Selbstkonfiguration) verfolgt. Ausgehend von einem aus der Planung und Konzeptionsphase entstandenen Modell der virtuellen Anlage (digitale Fabrik) wird dieses über speziell entwickelte Mechanismen mit der realen Anlage abgeglichen. Änderungen können dabei in beide Richtungen erfasst, verfolgt und die entsprechenden Systeme (übergeordnetes Leitsystem, reale und virtuelle Anlage) softwareseitig angepasst werden. (Schleipen et al. 2009; Baumann et al. 2009; Ebel et al. 2007)

SIARAS (Skill-based Inspection and Assembly of Reconfigurable Automation Systems) beschreibt Fähigkeiten, sogenannte „Skills“, von Geräten in einer abstrakten Form. Durch dieses Konzept der wissensbasierten Fertigung soll die Rekonfiguration erleichtert werden. Dabei steht nicht die Kommunikation im Fokus, sondern vielmehr die Interaktion von Produktionseinheiten, um eine spezifische Aufgabe zu lösen. Diese Skills werden mit einer durchzuführenden Aufgabe (Task), die zuvor beschrieben wurde, abgeglichen. Die Skill-Task-Verbindung ermöglicht es, die Schnittstelle zwischen Anwendung und Geräten zu spezifizieren. SIARAS fokussiert damit die Anwendungsprogrammierung auf Steuerungsebene. (Malec et al. 2007; Bengel 2008; Bengel & Pflüger 2008)

Beschreibungsformen und modellbasierte Ansätze abstrahieren die Informationen auf eine Metaebene. Sie erlauben so eine flexiblere Gestaltung der Schnittstellen. Es ist jedoch immer erforderlich, dass die Kommunikationspartner einheitliche Syntax, Semantik und Kommunikationsprotokolle verwenden.

3.3.4 Middleware und alternative Systemarchitekturen

Neben der Standardisierung und den Beschreibungsformen wird der Herausforderung der Konfiguration mit neuen bzw. veränderten System- oder Kommunikationsarchitekturen, meist unter der Verwendung einer Middleware (zu deutsch „Vermittlungssoftware“), begegnet. Ziel ist es, der Applikationsschicht

Dienste zur Verfügung zu stellen, mit denen entweder die Systemkommunikation erfolgen kann oder die Konfiguration erleichtert wird. Dabei ist es nicht zwingend erforderlich, die Kommunikation zu standardisieren. Es ist ausreichend, die verfügbaren Dienste zu vereinheitlichen. Wie bei Standards ist die Systemsoftware zu befähigen, die jeweiligen Beschreibungen zu interpretieren.

OSACA (Open System Architecture for Controls within Automation Systems), eine europäische Initiative, zielt auf eine herstellerunabhängige, offene Steuerungsarchitektur ab. Abbildung 21 zeigt die in dem Projekt entwickelte Steuerungsarchitektur. (I*PROMS 2006; OSACA 1996)

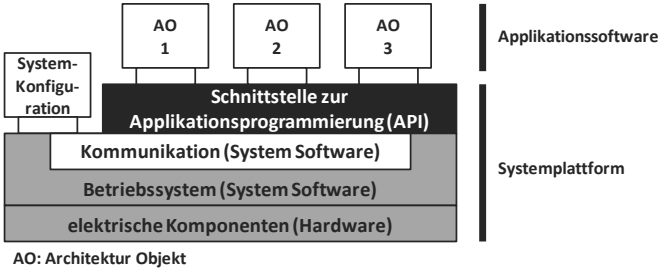

Abbildung 21: Steuerungsarchitektur von OSACA *(1996)*

Diese besteht aus einem Hardwareteil (z. B. elektronische Komponenten), einem Softwareteil (z. B. Betriebs- oder Kommunikationssystem) mit einer definierten Schnittstelle für sogenannte *Architektur Objekte* (AO), die als Applikationsmodule, also als Teil der Applikationssoftware, verstanden werden. Diese Softwareplattform stellt gewisse Services über definierte Programmierschnittstellen (API) zur Verfügung. Diese API ist die einzige Schnittstelle, über welche AOs Zugang zu der Plattform haben. So werden die unterliegenden Schichten vom System gekapselt und damit die Herstellung der Kommunikation vom Applikationsprogrammierer ferngehalten. Die eigentliche Systemkonfiguration (z. B. Topologie) muss der Plattform über eine extern generierte ASCII-Datei zur Verfügung gestellt werden. (I*PROMS 2006; OSACA 1996)
HÜMNOS, ein Folgeprojekt zu OSACA, stellt darüber hinaus Werkzeuge zur Erstellung der Konfigurationsdatei sowie zur Erzeugung der AOs zur Verfügung. (Pritschow 1997)

OSACA und HÜMNOS erleichtern durch den gewählten Ansatz die Interoperabilität zwischen Geräten unterschiedlicher Hersteller. Die Konfiguration sowie die AOs sind, soweit sie nicht vereinheitlicht sind, für jede Anwendung manuell neu zu erstellen.

Weber (2007) beschreibt ein Konzept für eine wandelbare Systemlandschaft und leitet daraus eine neue Softwarearchitektur für auf Wandelbarkeit ausgerichtete Transportsysteme ab. Das Konzept beruht auf einem streng hierarchischen Aufbau des Gesamtsystems und der Subsysteme. Es setzt voraus, dass alle Module auf allen Hierarchieebenen kommunikationstechnisch miteinander verbunden sind.
Neben einem übergreifenden Adressierungs- und Identitätsschema der Komponenten ist eine Kommunikationsmiddleware wesentlicher Bestandteil des Umsetzungskonzepts. Diese kontrolliert nach einem Publisher-Suscriber-Modell die Kommunikation der einzelnen Partner. Durch die Middleware werden, wie bei OSACA, Spezifika der unteren Kommunikationsschichten gekapselt. Damit ist eine Vereinheitlichung von Diensten zur Applikationsebene möglich. Im Falle der Umorganisation der Transportsysteme in einer Produktion, bspw. das Kappen einer Verbindung und das Herstellen einer anderen Verbindung, werden die Informationsflüsse automatisch umgeleitet und ggf. ein neuer Master bestimmt. (Feldmann et al. 2007a, b; Weber 2007)
Durch die funktionale Beschränkung auf Transportsysteme und die radikale Veränderung der Architektur ist zwar die Anwendbarkeit eingeschränkt, für die in Weber (2007) beschriebenen Fälle jedoch unterstützt die Architektur die Wandlungsfähigkeit erheblich.

Pöschmann & Krogel (2000) beschreiben ein Autoconfiguration Management Framework für Feldbusse (ACFG). Dies soll die Inbetriebnahme vorkonfigurierter Geräte erleichtern.
Das Framework ist durch eine Architektur gekennzeichnet, welche kommunikationsorientierte und anwendungsspezifische Dienste zur Verfügung stellt. Diese ermöglichen es, Geräteeinstellungen zu identifizieren und zu implementieren. So soll die bitweise Kommunikation zwischen den Geräten automatisiert hergestellt werden.
Die Umsetzung erfordert eine Definition der Kommunikationsschichten auf beiden Seiten. Zu dem Zeitpunkt der Veröffentlichung war eine Integration in den Profibus-Standard geplant. (Pöschmann & Krogel 2000)

Das vom BMBF geförderte Projekt PAPAS (Plug and Play Antriebs- und Steuerungskonzepte für die Produktion von Morgen) begegnete der Herausforderung ein Plug&Play-Konzept für Leichtbauroboter zu entwickeln mit der gezielten Anpassung des Protokollstacks bei der Ethernet-basierten Echtzeitkommunikation. Im Projekt wurde ein Schichtenmodell aufgezeigt, welches den Protokollstack sowohl des Masters als auch der Slaves in drei wesentliche Schichten unterteilt (vgl. Abbildung 22): Funktionsschicht (Functional Layer), Geräteschicht (Device Layer) und Busschicht (Bus Layer). (Grunwald et al. 2008; Plank et al. 2006)

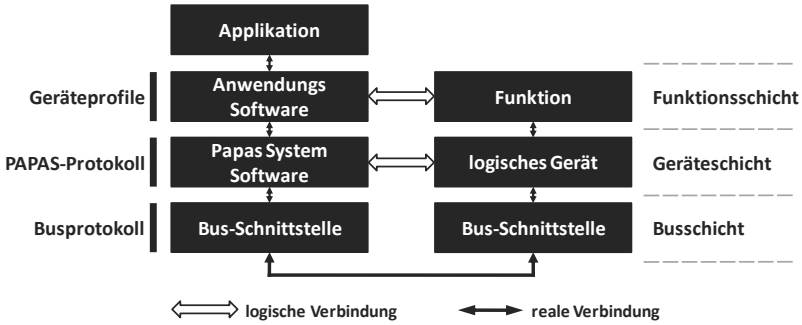

Abbildung 22: PAPAS-Schichtenmodell nach Plank et al. *(2006)*

Die *Funktionsschicht* wird dabei über Geräteprofile der *Geräteschicht* durch ein speziell entwickeltes PAPAS-Protokoll repräsentiert. Die *Busschicht* bleibt unberührt. Das PAPAS-Protokoll bietet verschiedene Dienste an, die eine Konfiguration der Echtzeitkommunikation ermöglichen. Beispiele hierfür sind die Dienste *SET_ADDRESS* – zur Zuweisung einer Geräteadresse oder *SET_FEATURE* – zur Aktivierung einer bestimmten Geräteeigenschaft. Über diese standardisierten Dienste des PAPAS-Protokolls kann der Master die Konfigurationsinformationen von angeschlossenen Geräten erhalten und die Kommunikation entsprechend während des Betriebs anpassen. (Grunwald et al. 2008; Plank et al. 2006)

Bei konformer Implementierung des PAPAS-Protokolls und der Software bei allen Kommunikationspartnern kann eine automatische Konfiguration vordefinierter Geräteklassen mit vordefinierter Funktionalität erreicht werden. Offen

bleibt, inwieweit die Integration dieser Protokolle in den Echtzeitkommunikationsstack die Verfügbarkeit des Systems beeinflusst. Zudem ist es nicht möglich, Geräte ohne Unterstützung des PAPAS-Protokolls zu integrieren.

Das Projekt SMErobot™ – eine von der Europäischen Union geförderte Robotikinitiative zur Stärkung der Wettbewerbsfähigkeit von kleinen und mittleren Produktionsbetrieben – fokussierte unter anderem die einfache Einrichtung und Programmierung von Industrierobotern. In diesem Teilprojekt wurde auf Basis der bereits beschriebenen Standards XIRP und UPnP™, welche die unteren Kommunikationsebenen bis zur Darstellungsschicht abdecken, eine serviceorientierte Steuerungsarchitektur erarbeitet. (Naumann et al. 2006, 2007; Verl & Naumann 2008b)
So soll über die standardisierten XIRP- und UPnP™-Schnittstellen auf Funktionalitäten der in der Roboterzelle vorhandenen Geräte zugegriffen werden können. Naumann et al. (2007) beziehen sich auf die Einteilung von Grunwald et al. (2008) in die Ebenen Kommunikation, Konfiguration, Applikation und ordnen die Arbeiten in die Applikationsebene ein. Abbildung 23 zeigt das Systemkonzept.

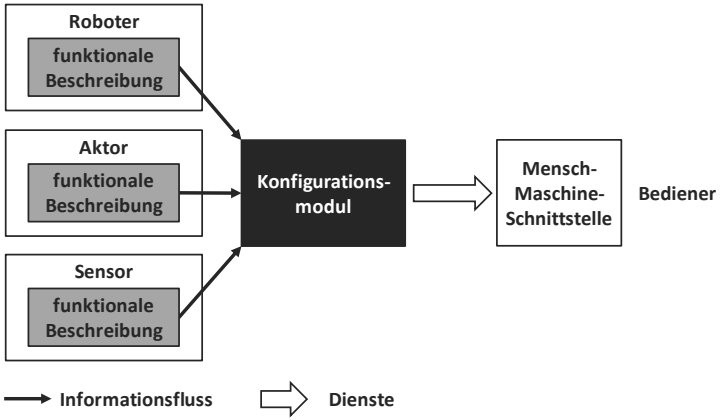

Abbildung 23: SMErobot™-Systemkonzept zur einfachen Programmierung nach Naumann et al. *(2006)*

Roboter und Peripheriegeräte stellen über ihre Gerätebeschreibung gleichermaßen dem Konfigurationsmodul, ein übergeordnetes System, ihre Dienste bereit.

Dies erfolgt, wie in Verl & Naumann (2008b) beispielhaft gezeigt, über einem Zustandsgraphen, welcher mit XIRP-Befehlen angereichert ist. Dieses Modul bestimmt zusammen mit einer Prozessbeschreibung die vom System ausführbaren Prozesse und stellt diese dem Anwender zur Verfügung. Nachdem der Nutzer die Anwendung erstellt hat, überführt das Modul diese in ausführbaren Code. (Verl & Naumann 2008b)
Das Vorgehen erlaubt die einfache Erzeugung einer Ablaufsequenz für ein Roboterprogramm. Durch die Verwendung des XIRP- bzw. UPnP™-Standards bleiben deren Einschränkungen jedoch bestehen, wie bspw. die nicht echtzeitfähige Kommunikation.

Der Einsatz von Middleware sowie von neuen Systemarchitekturen erlaubt im Gegensatz zur reinen Standardisierung einen flexibleren Umgang, vor allem mit unterschiedlichen hardwarenahen Protokoll- und Gerätearchitekturen. Dies erfolgt durch eine Kapselung der unteren Kommunikationsebenen und der Bereitstellung von Services. So genügt eine Vereinheitlichung dieser Ebenen. Jedoch werden wechselnde Funktionalitäten sowie die Echtzeitfähigkeit nicht in vollem Umfang unterstützt. Auch erfordern Middlewarekonzepte sowie alternative Systemarchitekturen eine einheitliche Implementierung der Middleware auf beiden Seiten der Kommunikation.

3.4 Situationsanalyse und Handlungsbedarf

Die bisherigen Ansätze zur Vereinfachung der Konfiguration sind in die vier Domänen Plug&Play, standardisierte Protokolle, Beschreibungen und Modelle und Middleware und Systemarchitektur eingeteilt. Zur Analyse des Standes der Forschung und Technik und zur Identifikation des Handlungsbedarfes wurden die beschriebenen Ansätze in einer Bewertungsmatrix (vgl. Tabelle 2) zusammengefasst.

Die Einteilung erfolgte nach dem inhaltlichen Fokus der Arbeiten sowie der Bewertung hinsichtlich der Relevanz für die automatische Konfiguration von Robotersystemen. Die folgenden sechs Kriterien – abgeleitet aus der Zielsetzung (vgl. Kapitel 1) – ordnen die Arbeiten ein und indizieren deren jeweilige Eignung:

○ nicht erfüllt
◐ teils erfüllt
● voll erfüllt
– Bewertung nicht möglich

	Plug& Play		standardisierte Protokolle					Beschreibung und Modelle				Middleware und Systemarchitekt.				
	Enumeration	Common Base Prot.	OPC	Weihenst. Standard	Leitsystem-Projekt.	ORiN	XIRP-Standard	Gerätebeschreibung	EmsA	Produflexil	SIARAS	OSACA	Wandelb. Systeme	ACFG	PAPAS	SMErobot
Betrachtungsraum Robotersystem	○	○	○	○	○	◐	●	◐	○	○	○	○	○	◐	●	●
heterogene Systemlandschaft	◐	◐	◐	○	●	◐	◐	○	○	○	●	●	●	○	○	○
Universalität bzgl. Gerätefunktional.	●	◐	◐	○	◐	○	◐	–	–	●	●	●	○	–	◐	◐
Echtzeitfähigkeit	○	○	○	○	○	○	○	●	○	○	○	◐	○	●	●	◐
Zielsystemunterstützung	○	○	○	○	○	○	◐	○	○	○	○	○	○	◐	◐	○
automatische Konfiguration	●	●	◐	◐	◐	○	◐	◐	◐	●	○	○	○	●	●	◐

Tabelle 2: Bewertung des Stands der Technik und Forschung

1. *Betrachtungsraum Robotersystem* – signalisiert den Fokus der Arbeiten auf Robotersysteme;

2. *Heterogene Systemlandschaft* – zeigt die Fähigkeit des Ansatzes, mit unterschiedlichen Systemen, Formaten und Standards umzugehen;

3. *Universalität bzgl. Gerätefunktionalitäten* – drückt die Fähigkeit aus, bei der Konfiguration mit bisher unbekannten Gerätefunktionen umzugehen;

4. *Echtzeitfähigkeit* – indiziert die Fähigkeit, harte Echtzeitanforderungen während des Produktivbetriebs zu gewährleisten;

5. *Zielsystemunterstützung* – beschreibt die Eigenschaft, Robotersysteme vollständig zu konfigurieren;

6. *Automatische Konfiguration* – Konfiguration ohne die Erfordernisse eines Benutzereingriffs im Sinne des Plug&Play-Gedanken bei der Inbetriebnahme.

Das erforderliche Expertenwissen sowie die Inbetriebnahmezeit stellen weitere relevante Kriterien dar, werden aber an dieser Stelle ausgespart. Die veröffentlichten Informationen lassen dafür keine objektive Bewertung zu.

Die Bewertungsmatrix verdeutlicht, dass bestehende Technologien aus dem Bereich Plug&Play der PC-Welt die Konfiguration vollständig automatisch durchführen können. Auch die Kapselung der Komplexität vor dem Benutzer zeichnet diese Systeme aus. Ein Übertrag dieser Technologie auf Robotersysteme ist aufgrund der mangelnden Fähigkeiten zur Echtzeitkommunikation nicht uneingeschränkt möglich. Außerdem werden an automatisierungstechnische Systeme weitergehende Anforderungen an Zuverlässigkeit und Verfügbarkeit gestellt. Lösungen im Office-Bereich können diese Anforderungen nicht, oder nur bedingt unterstützen, liefern aber die Grundlage für neue Methoden.

Standardisierte Protokolle ermöglichen einen einfach einzurichtenden Datenaustausch. Die Unterstützung von geräteindividuellen Funktionen ist dabei nicht gegeben. Die Interoperabilität mit der heterogenen Systemlandschaft wird durch eine teils aufwendige Implementierung der Standards in die individuellen Systeme gelöst. Eine automatische Konfiguration kann nicht durch Standards alleine realisiert werden. Es bedarf immer eines Betriebssystems, welches die Potenziale der Standards nutzt, um die Konfiguration zu vereinfachen bzw. zu automatisieren.

Beschreibungen und Modelle haben durch die semantische Abstraktion ihre Vorteile in der Universalität. So lassen sich Schnittstellen beschreiben, anstatt sie festzulegen. Es bedarf immer einer Implementierung in den jeweiligen Systemen, die den Ablauf und Umgang mit den Modellen und Beschreibungen durchführt. Modelle bzw. Beschreibungen und Realität müssen miteinander übereinstimmen, um eine gemeinsame Kommunikation zu ermöglichen. Die aggregierten Modelle aktuell zu halten, ist eine zeitaufwendige Herausforderung.

Middleware und neue Systemarchitekturen erlauben eine Abstraktion der Kommunikationsebene. Es können also Informationen unabhängig von der Übertragung und Systemgestaltung übertragen werden. Die freie Definition der Schnittstellen, um individuelle Funktionalitäten abzubilden, wird dabei aufgegeben.

Automatische Konfiguration (Plug&Produce) ist mehr als die Gestaltung und Normierung einheitlicher Schnittstellen. Durch diese können lediglich vorge-

dachte Gerätefunktionalitäten abgedeckt werden. Erst das Zusammenspiel von – zumindest bis zu einem gewissen Grad – einheitlichen Schnittstellen und einer für das Anwendungsgebiet angepassten Methodik zur automatischen Konfiguration führen zu einem effizienten Plug&Produce-Ansatz. Bei bestehenden Herangehensweisen fehlen umfassende und übergreifende Anforderungen, speziell an Robotersysteme.

Es bedarf der Entwicklung einer Methodik, die eine grundlegende Vereinfachung bzw. Automatisierung der Konfiguration und Inbetriebnahme von Robotersystemen ermöglicht. Derzeit fehlen Grundlagen sowie Methoden und Verfahren, durch welche die limitierenden Faktoren überwunden werden können. Bestehende Ansätze in Kombination mit neuartigen Methoden stellen dabei eine vielversprechende Vorgehensweise dar. So müssen die Vorteile bestehender Ansätze sinnvoll miteinander kombiniert werden, um neue Lösungen zu generieren. Der Handlungsbedarf liegt auch in der Analyse und Klassifizierung der für die Planung, die Systemintegration und den Betrieb von Robotersystemen erforderlichen Informationen, der gerätetechnischen Strukturierung der Peripheriekomponenten und der Entwicklung von Modellen, die eine Beschreibung des Robotersystems zulassen. Da Industrieroboter in den relevanten Anwendungsfällen häufig für die Steuerung der gesamten Automatisierungszelle eingesetzt werden, ist die Erforschung einfacher und intuitiver Methoden zur ganzheitlichen, automatischen Rekonfiguration von Robotersystemen auf der Basis der Robotersteuerung ein zentraler Aspekt. Dabei soll die Konfiguration die Nutzung bestehender und bewährter Programmiermethoden (z. B. Online- und Offline-Programmierung) gängiger Robotersteuerungen unterstützen.

4 Anforderungsanalyse

4.1 Allgemeines

Für die Gestaltung von Plug&Produce-Robotersystemen bedarf es einer detaillierten Analyse der Anforderungen. Zu deren Spezifikation ist eine umfassende Kenntnis über die Informationsflüsse in einer Roboterzelle, sowohl während der konventionellen Konfiguration als auch während des produktiven Betriebs, erforderlich. Dieses Kapitel beschreibt die durchgeführte Informationsflussanalyse und deren Ergebnisse. Aus diesen Ergebnissen wird zusammen mit allgemeinen Erfordernissen ein detailliertes Bild hinsichtlich der technischen, nutzerorientierten, aber auch wirtschaftlichen Anforderungen erstellt.

4.2 Informationsflussanalyse

4.2.1 Vorgehensweise zur Informationsflussanalyse

Die Informationsflussanalyse hat zum Ziel, die in einer Roboterzelle auftretenden Informationsflüsse zu untersuchen und zu klassifizieren. Die Diversität der Geräte und der Software mit ihren unterschiedlichen Informations- und Datenstrukturen wurde ebenso betrachtet, wie die heterogenen Übertragungswege. So fand nicht nur die elektronische Datenübertragung Berücksichtigung, sondern auch die von Hand ausgeführte Informationsübermittlung zur Konfiguration der Robotersteuerung und der Programmierumgebung, welche durch Systemintegratoren während des Systemaufbaus vorgenommen wird. Wichtig sind dabei zudem mögliche Abhängigkeiten und Freiheitsgrade hinsichtlich der Dateninterpretation. Abbildung 24 zeigt das gewählte dreistufige Analysevorgehen. (Reinhart & Krug 2010)

1. Aus der Menge aller möglichen Systeme wurden anhand gängiger Anwendungen und bestehender Komponentenstrukturen repräsentative Anwendungsfälle abgeleitet.

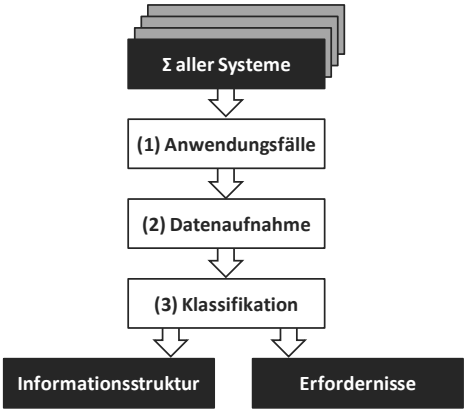

Abbildung 24: Vorgehen zur Informationsflussanalyse

2. Für jeden Anwendungsfall wurden an den jeweiligen Systemgrenzen der Geräte virtuelle Schnitte durch das Robotersystem gezogen und der Informationsfluss an diesen Schnittstellen in einer Datensammlung festgehalten.

3. Schließlich wurden die erhobenen Informationen klassifiziert und eine Informationsstruktur abgeleitet.

4.2.2 Auswahl und Analyse repräsentativer Anwendungsfälle

Die vorherrschende Gerätevielfalt und ihre beinahe unbegrenzten Kombinationsmöglichkeiten in einer Roboterzelle machen eine Berücksichtigung aller möglichen Konstellationen nahezu unmöglich. Anwendungsfälle stellen eine bewährte Methode dar, um Anforderungen aufzunehmen und einen Großteil der relevanten Aspekte abzudecken (Bittner & Spence 2003). Zur Auswahl geeigneter Anwendungsfälle wurden zwei wesentliche Kriterien berücksichtigt (Reinhart & Krug 2010):
Gängige Anwendungen von Industrierobotern – Nach IFR (2011) werden die meisten Roboter für die Handhabung und das Schweißen eingesetzt.
Peripheriegerätekategorien – Peripheriegeräte lassen sich in die Kategorien Mess- und Testgeräte, Sicherheitssysteme, Spannvorrichtungen, Systeme zur

Positions- und Lageerkennung, Zuführ- und Positioniereinrichtungen und prozessspezifische Peripheriegeräte unterteilen.

Anhand dieser Kriterien wurden *Handhabung und Montage* und *Schweißen* als geeignete Anwendungsfälle definiert. Diese Szenarien decken je einen Hauptanwendungsfall und mindestens ein charakteristisches Gerät jeder Kategorie ab. Die Analyse erfolgte dabei einerseits im ursprünglichen Konfigurationszustand, andererseits während der Rekonfiguration. Sie wurde auf der Informationsbasis realer Anlagenkomponenten durchgeführt. (Reinhart & Krug 2010)

Anwendungsfall – Handhabung und Montage

Ausgangsszenario – Dieser Anwendungsfall beinhaltet eine Roboterzelle, in der ein Bauteil von einem Förderband mit einem pneumatischen Greifer aufgenommen und anschließend montiert wird. Die Baugruppe wird auf einem weiteren Förderband abgelegt. Die Zelle besteht somit neben dem Roboter aus zwei Förderbändern, einem pneumatischen Greifer und einer Spannvorrichtung für die Montage (vgl. Abbildung 25 links).

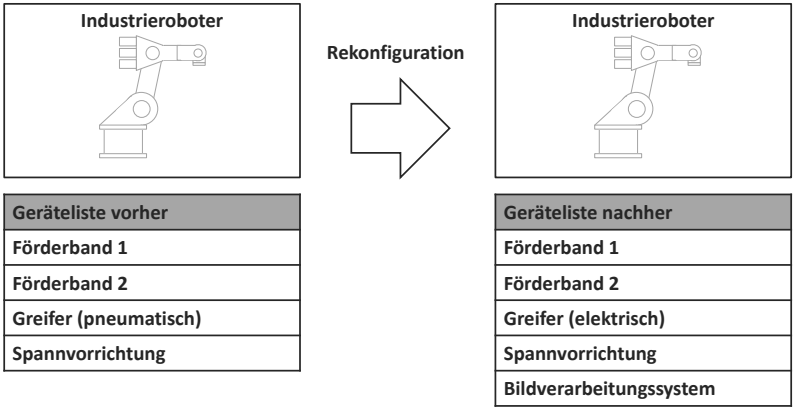

Abbildung 25: Anwendungsfall Handhabung und Montage vor (links) und nach (rechts) der Rekonfiguration

Die Geräte sind über ein Industrial-Ethernet-Bussystem miteinander verbunden. Die Ventile und Sensoren des pneumatischen Greifers sind mit einer Ein- und Ausgangsbaugruppe des Bussystems verknüpft.

Rekonfiguriertes Szenario – Die Einführung eines weiteren Produktes mit leicht unterschiedlichen Abmessungen erfordert einen elektrischen Greifer. Zur Qualitätssicherung wird ein Bildverarbeitungssystem installiert (vgl. Abbildung 25 rechts).

Anwendungsfall – Schweißen

Ausgangsszenario – Im Anwendungsfall *Schweißen* ist die Roboterzelle dafür ausgelegt, flächige Metallbauteile miteinander zu verschweißen. Die Zelle beinhaltet ein Schweißgerät mit einem Schweißbrenner, eine Spannvorrichtung und eine Reinigungsstation für den Schweißbrenner (vgl. Abbildung 26 links). Alle Geräte sind, wie im ersten Beispiel, ebenfalls über ein Bussystem miteinander verbunden.

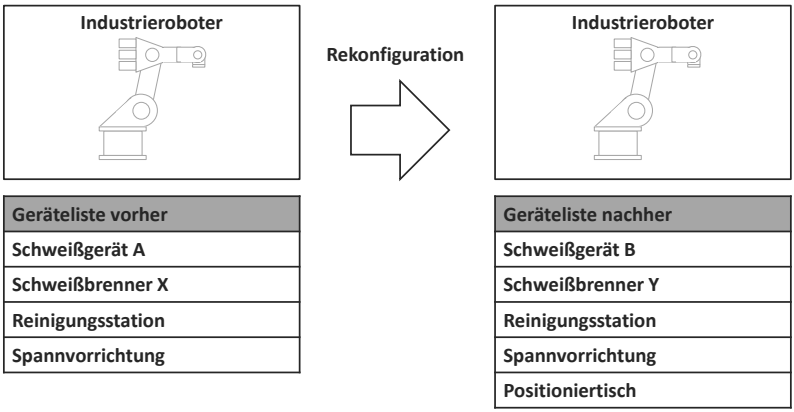

Abbildung 26: Anwendungsfall Schweißen vor (links) und nach (rechts) der Rekonfiguration

Rekonfiguriertes Szenario – Aufgrund von Änderungen an der Schweißnaht des Produktes muss das Schweißgerät ersetzt werden und ein Zwei-Achs-Positioniertisch wird in die Zelle integriert (vgl. Abbildung 26 rechts).

4.2.3 Analyse und Datensammlung

Zur Aufnahme der Daten wurden virtuelle Schnitte durch die Roboterzelle auf informationstechnischer Ebene gelegt. Analog zum „Freischneiden“ in der Mechanik, wurden die Verbindungen und Interaktionen an den Schnittstellen betrachtet. Die „Schnitte“ wurden anhand der Modulgrenzen in der Zelle gezogen. Zur Datenaufnahme wurden folgende Größen als Akquisitionsschema verwendet: Echtzeitanforderungen, Informationsursprung und -ziel, Kommunikationsebene, Datengröße, Datenformat, Informationstyp, Übertragungsweg, Sicherheitsrelevanz, Freiheitsgrade in Dateninterpretation oder Festlegung. Das Ergebnis ist eine Datenmatrix der übermittelten Informationen mit den Charakteristika der einzelnen Informationen nach dem Konfigurationsschema. (Reinhart & Krug 2010)

4.2.4 Klassifikation und Informationsstruktur

Anhand der festgestellten Abhängigkeiten im Informationsfluss ergibt sich eine erste Einordnung nach dem Kommunikations- und Konfigurationsniveau (Reinhart & Krug 2010). Das bedeutet eine Einordnung in die Ebenen Kommunikation, Interpretation und Systeminformation, Applikation und Programmfunktion.

Kommunikation – Kommunikationsdaten bestehen hauptsächlich aus echtzeitkritischen Prozessdaten sowie bussystemabhängigen Netzwerkmanagementinformationen während der Betriebsphase. Datenformate sind einfache *Bool*-, *Integer*-, *Real*- oder *Double*-Werte, deren Übertragung ausschließlich elektronisch über das Bussystem stattfindet. Darüber hinaus werden Einstellungen des Bussystems manuell übertragen. Dies sind Zykluszeiten, zulässige Latenzzeiten, Knotennummern, etc.

Interpretation und Systeminformationen – Informationen über die Interpretation der Prozessdaten im Bussystem, der Robotersteuerung und der Programmierumgebung sowie spezifische Geräteinformationen, z. B. die geometrische Gestalt der Geräte, werden manuell übermittelt oder zumindest deren Transport manuell angestoßen. Dementsprechend bestehen Abhängigkeiten zu den Prozessdaten und zur gesamten Systemkonstellation. Die erforderlichen Infor-

mationen stammen meist aus Datenblättern oder aus Bussystem-spezifischen Beschreibungsdateien.

Applikation und Programmfunktion – Diese Informationen definieren die Schnittstelle des Systems zur Applikation auf der einen und die Beschreibung der Gerätefunktionen für das Applikationsprogramm auf der anderen Seite. Hier werden durch implizites Wissen und Erfahrung des Bedieners – meist des Systemintegrators – sowie durch Informationen aus Handbüchern die Gerätefunktionalitäten in Form von Unterprogrammen zu der Programmierumgebungen übertragen. Diese Unterprogramme enthalten Ein-/Ausgangsinstruktionen, die wiederum von der Konfiguration des Bussystems abhängen.

4.2.5 Schlussfolgerungen für die Konfiguration

Die Ergebnisse der Informationsanalyse erlauben eine Ableitung von Erfordernissen für den Rekonfigurationsprozess (Reinhart & Krug 2010).

Erfordernisse der Datenverfügbarkeit – Die für die Konfiguration erforderlichen Informationen müssen in einer maschinenlesbaren Form vorhanden sein. Nachdem sogar vollständig unbekannte Geräte an das System angeschlossen werden sollen, ist diese Information auf den Geräten zu hinterlegen.

Erfordernisse des Kommunikationssystems – Es muss möglich sein, mit dem Kommunikationssystem echtzeitkritische Daten ebenso wie große Datenmengen zu übertragen. Industrial-Ethernet-Bussysteme erfüllen bspw. diese Anforderungen.

Erfordernisse des automatischen Konfigurationsprozesses – In einigen Fällen muss der Bediener Entscheidungen über bestimmte Konfigurationsaspekte treffen, wie z. B. die Ein-/Ausgangszuordnung in der Robotersteuerung. Diese Entscheidungen müssen von dem automatischen Konfigurationsprozess übernommen werden. Auch das eventuelle Vorhandensein unterschiedlicher Datenformate zur Informationsbereitstellung muss Berücksichtigung finden. Einige Setup-spezifische Informationen, wie die geometrische Position eines Gerätes, sind nicht im Vorhinein datentechnisch zu erfassen. Für diese Fälle müssen Möglichkeiten geschaffen werden, diese Informationen auf eine einfache Weise zu hinterlegen.

Erfordernisse an die bereitzustellenden Informationen – Die Ergebnisse der Informationsflussanalyse geben nicht nur Aufschluss über den anzustrebenden Konfigurationsablauf. Sie definieren auch die Informationskategorien, welche bei der Konfiguration der betrachteten Zielsysteme erforderlich sind. Sie lassen sich in die Kategorien (1) Hilfe, (2) Allgemeine Informationen, (3) Geometriebeschreibung, (4) Funktionsbeschreibung, (5) Prozessdaten Mapping und (6) Echtzeitkommunikation einteilen.

Diese Erfordernisse tragen zu einer umfassenden Definition der Anforderungen an Plug&Produce-Robotersysteme bei.

4.3 Anforderungen an Plug&Produce-Robotersysteme

4.3.1 Zielkriterien von Plug&Produce-Robotersystemen

Das Plug&Produce-Prinzip im Bereich industrieller Robotersysteme hat die Reduzierung des Konfigurationsaufwands für die Bearbeitung neuer Aufgaben zum Ziel. Im Zuge dessen sind drei maßgebliche Zielkriterien zu erfüllen, welche in einem Zielkonflikt zueinanderstehen: Effizienz, Universalität und Einfachheit.

Effizienz – Im Kontext der automatischen Konfiguration stellt die Effizienz eine Kenngröße des Konfigurationsnutzens im Verhältnis zum Zeit- und Kostenaufwand für die Konfiguration dar. Besonders unter der Maßgabe kurzer Einsatzzeiten einer Systemkonfiguration – bedingt durch kleine Losgrößen – und damit häufiger Konfigurationsvorgänge, ist eine zeit- und kostenminimale Umrüstung essenziell. So können Roboter-Stillstandszeiten reduziert und der konfigurationsbedingte Kostenanteil der Produktion gesenkt werden.

Universalität – Der Einsatz von Industrierobotern für unterschiedlichste Aufgaben bedingt eine fortwährend wachsende Diversität an Herstellern, Geräten und Gerätefunktionen, Kommunikationssystemen und branchen- bzw. anwenderspezifischen Standards. Die Anwendbarkeit der zu entwickelnden Plug&Produce-Methode auf die unterschiedlichen Systeme, Funktionen und Einsatzgebiete steht für den Erfüllungsgrad der Universalität.

Einfachheit der Durchführung – Dieses Zielkriterium stellt ein Maß der Abstraktion komplexer Vorgänge der Konfiguration gegenüber dem Bediener dar. Dies

bedeutet, dass Dauer, Qualität und Ergebnis der Konfiguration weitestgehend unabhängig vom systemtechnischen Wissen und Erfahrung der durchführenden Person sind.

Die benannten Zielkriterien sowie industrielle Rahmenbedingungen – z. B. industrielle Umgebungen, Einsatzbedingungen und -felder, bestehende technische Lösungen, Personal für die Anlagenbedienung und andere Einflussfaktoren – stellen Anforderungen an Plug&Produce-Robotersysteme. Diese sind in die drei Gruppen *technische Anforderungen*, *nutzerorientierte Anforderungen* und *wirtschaftliche Anforderungen* gegliedert.

4.3.2 Technische Anforderungen

Induziert durch bestehende Systemarchitekturen, technische Lösungen, verfügbare Roboterstrukturen und die durchgeführte Informationsflussanalyse, ergeben sich allgemeine und spezifische technische Anforderungen.

Allgemeine technische Anforderungen

Die Konfiguration ist maßgeblich durch die eingesetzten Roboter, Geräte und die zu erfüllende Aufgabe geprägt. Ihre Diversität hat zur Folge, dass sich keine allgemeingültigen, quantitativen Anforderungen hinsichtlich der Leistungsfähigkeit und des Funktionsumfangs des Konfigurationsergebnisses beschreiben lassen. Der Konfigurationsvorgang selbst muss jedoch an bestehende und zukünftige prozessspezifische Gegebenheiten angepasst werden können. Grundsätzlich sind bei der Gestaltung folgende Aspekte zu berücksichtigen, die sich teilweise aus der Domäne rekonfigurierbarer Steuerungen und Anlagen ableiten lassen (I*PROMS 2006):

Systemtechnische Gesamtzuverlässigkeit – Der Zuverlässigkeit von Produktionssystemen kommt eine besondere Bedeutung zu, da dadurch die Erfüllung der Produktionspläne sichergestellt wird. Ein automatischer Konfigurationsprozess muss daher so gestaltet werden, dass die Zuverlässigkeit des Gesamtsystems während des Betriebs durch den Konfigurationsvorgang nicht verringert wird.

Echtzeitfähigkeit – Bspw. erfordert die Synchronisation von Antrieben oder Bewegungen eine Echtzeitkommunikation der Geräte in einer Roboterzelle. Diese

darf während des Betriebs durch die Fähigkeit der automatischen Konfiguration nicht beeinträchtigt werden.

Funktionsumfang – Geräte, die in der Roboterzelle zum Einsatz kommen, verfügen in der Regel über individuelle Funktionsumfänge. Die Konfiguration muss die Integration von Geräten mit ihrer vollen Funktionalität unterstützen.

Flexibilität – Die Flexibilität von Industrierobotern darf durch die Konfiguration nicht reduziert werden. Einschränkungen durch Unterstützung einer beschränkten Prozess- oder Anwendungsbandbreite reduzieren den Mehrwert der universell einsetzbaren Bewegungsautomaten.

Kompatibilität – Um die Kompatibilität zwischen verschiedenen Geräten, Robotern und Kommunikationssystemen sicherzustellen, ist eine Konformität zu den unterschiedlichen bestehenden Standards einzuhalten. Auch Geräte ohne Plug&Produce-Fähigkeiten müssen in das Gesamtsystem integriert werden können.

Sicherheit – Die Sicherheit muss nach geltenden Normen und Richtlinien zu jeder Zeit gewährleistet sein.

Roboter – Die hoch diversen Charakteristika der eingesetzten Industrieroboter hinsichtlich Systemgestaltung, Programmiersprachen, Schnittstellen und Konfigurationsumgebung sind bei der Gestaltung zu berücksichtigen.

Kommunikationssystem – Der Trend heterogener Feldbusprotokolle und Kommunikationssysteme setzt sich im Bereich des Industrial-Ethernet weiter fort. Eine Übertragbarkeit und im besten Fall eine Unabhängigkeit hinsichtlich des verwendeten Industrial-Ethernet-Systems ist anzustreben.

Spezifische Konfigurationsbedarfe der betrachteten Zielsysteme

Neben den allgemeinen Anforderungen besitzen die zu konfigurierenden Systeme – Robotersteuerungen, Programmierumgebungen und die Dokumentation – spezifische technische Anforderungen.

Robotersteuerung – Abbildung 27 zeigt die Schichten des Protokollstacks eines Industrieroboters. Die gekennzeichneten Schichten müssen in Abhängigkeit von den angeschlossenen Geräten erstellt werden und sind daher individuell zu konfigurieren. Zur Gewährleistung der vollständigen Funktionalität dieser

Geräte ist auch eine Einbindung auf Applikationsebene erforderlich. So stehen in der Robotersteuerung die einzelnen Gerätefunktionen zur Verfügung. Die Konfiguration muss für verschiedenste Industrieroboter möglich sein.

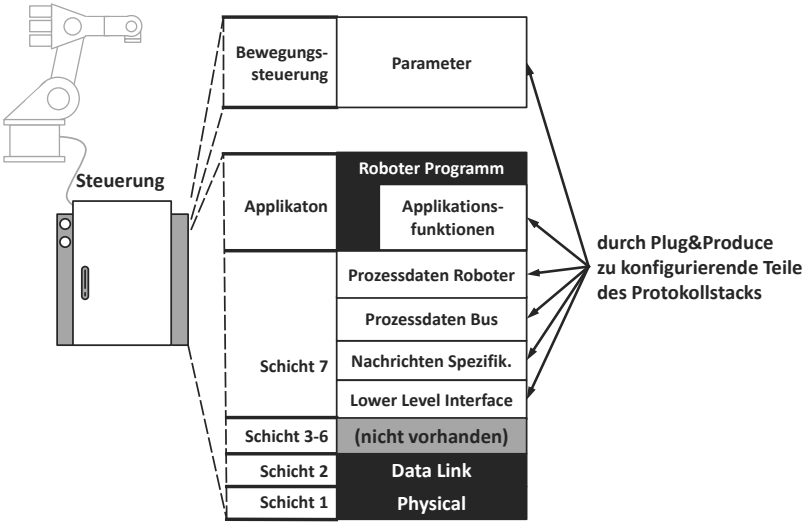

Abbildung 27: Zu konfigurierende Schichten des Protokollstacks eines Industrieroboters

Programmierumgebung – Bei der Teach-In-Programmierung müssen die Gerätefunktionalitäten mit in die Programmierumgebung eingebunden werden, sodass mit Funktionsaufrufen (z. B. *greifer_oeffnen()*) die Geräte gesteuert werden können. Im Fall der simulativen Offline-Programmierung sind zudem Signalverläufe, die geometrische Anordnung der Komponenten und logische Zusammenhänge durch die automatische Konfiguration zu unterstützen.

Dokumentation – Die Erstellung einer ausreichenden Dokumentation des Robotersystems nach unterschiedlichen und teilweise auch unternehmensspezifischen Richtlinien (z. B. VDI 4500 Blatt1: Technische Dokumentation – Benutzerdokumentation) muss durch die automatische Konfiguration unterstützt werden.

Anforderungen an den Konfigurationsprozess

Rekonfigurationsprozesse im PC sowie im produktionstechnischen Bereich lassen sich nach I*PROMS (2006) und Grunwald et al. (2008) in drei unterschiedliche Klassen unterteilen:

static/cold – Die statische Konfiguration beschreibt den Systemwandel während des Stillstands des Systems.

online/coordinated – Konfigurationsänderungen sind während der Hochlaufphase möglich.

dynamic/hot – Änderungen können während des laufenden Produktivbetriebs durchgeführt werden.

Im Bereich der Industrieroboter wird die Veränderung der Konfiguration meist durch einen Wechsel der Produktionsaufgabe induziert. Das bedeutet, dass nach einem Konfigurationswechsel das Anwendungsprogramm erstellt, angepasst oder zumindest getestet werden muss. Hierfür muss in jedem Fall der Produktionsbetrieb unterbrochen werden. Eine statische bzw. online-Unterstützung der Konfiguration ist daher für Robotersysteme ausreichend.

Grundvoraussetzung für eine vereinfachte bzw. automatische Konfiguration ist die *Abstraktion* der Systemkomplexität und der Kommunikationsdetails gegenüber dem Nutzer. Diese Abstraktion kann im Bezug auf den Stand der Technik auf zweierlei Weise erfolgen:

Abstraktion durch Standardisierung der Kommunikationsprotokolle und Klassifizierung der Systemkomponenten – Standardisierung hat den Vorteil einer einfachen Struktur und Wiederverwendbarkeit der Protokolle. Nachteilig sind jedoch die Starrheit gegenüber Veränderungen im Funktionsumfang, die Weiterentwicklung bestehender Standards und die Notwendigkeit der zwingend einheitlichen Nutzung des standardisierten Protokolls.

Abstraktion durch Metainformationen der Kommunikationsprotokolle und Systemkomponenten – Informationen über die Gerätebeschaffenheit und deren Kommunikationsschnittstellen ermöglichen eine individuelle, geräte- und systemunabhängige Abstraktion. Dafür ist jedoch ein komplexerer Konfigurationsvorgang erforderlich.

4.3.3 Nutzerorientierte Anforderungen

Das System muss an die Anforderungen des Bedieners angepasst sein. Ähnlich wie beim PC sollen auch Roboterlaien in die Lage versetzt werden, die Systeme umzurüsten. Der Anwender soll sich auf den Prozess – Kleben, Schweißen, etc. – konzentrieren und nicht mit der systemischen Komplexität belastet werden. Die erforderliche Erfahrung und die Kenntnis über den Konfigurationsvorgang sollten daher an den Bediener vor Ort angepasst sein. Die Einfachheit der Bedienung und die Abstraktion der Konfigurationsdetails müssen daher im Vordergrund stehen. Der Nutzer soll kein kommunikations-, bus- oder roboterspezifisches Wissen für die Konfiguration einbringen.

Dadurch werden auch kleinen Unternehmen, für welche die Beschäftigung eines Roboterexperten nicht lohnt oder nicht finanzierbar ist, in die Lage versetzt selbst Roboterlösungen einzusetzen. Der kostenträchtige Einsatz von Experten oder von externen Systemintegratoren kann reduziert oder sogar vermieden werden.

4.3.4 Wirtschaftliche Anforderungen

Für ein Plug&Produce-fähiges Robotersystem sind einmalige Aufwände der Hersteller – Roboterhersteller und Gerätehersteller – erforderlich, um diese Funktionalität in ihre Geräte zu integrieren. Um einen wirtschaftlichen und nachhaltigen Einsatz von Plug&Produce-Robotersystemen zu gewährleisten, muss der Mehrwert, der dem Anwender durch einen geringeren Konfigurationsaufwand entsteht, größer sein, als die Summe der Aufwände der Hersteller. Daher sind diese gering zu halten und der Nutzen ist zu maximieren. Der Nutzen ist abhängig von der Häufigkeit der Konfiguration, dem Wegfall von speziell geschultem Personal bzw. externen Fachkräften und dem zeitlichen Einsparpotenzial im Einzelfall. Der einmalige Aufwand der Hersteller zur Befähigung der Geräte spielt im Vergleich zu der Häufigkeit des Konfigurationsvorgangs für den Kunden der Geräte eine weniger signifikante Rolle.

5 Gestaltung von Plug&Produce-Modulen

5.1 Allgemeines

Die Fähigkeit der beliebigen Integration bzw. Entfernung von Komponenten aus einem roboterbasierten Systemverbund hat gewisse Auswirkungen auf die erforderliche mechatronische Systemstruktur. Eine zwingende strukturelle Eigenschaft, die durch die bestehenden Trends wie steuerungstechnische Dezentralisierung erfüllt wird, ist ein gewisser Grad an Modularität der Systemkomponenten. Dies bedeutet, es herrscht eine Abgeschlossenheit und auch Autarkie der Peripheriegeräte untereinander und gegenüber dem Roboter. Die nächsten Abschnitte geben einen Überblick über technische Systeme nach Ropohl (2009) und leiten daraus ein Modularisierungskonzept für Robotersysteme ab. Ein Vorgehen zur Modularisierung wird aufgezeigt, welches Handlungsanweisungen zur Erstellung von Plug&Produce-fähigen Modulen gibt.

5.2 Systeme und Module

Sachsysteme – Systeme im technischen Kontext – sind als nutzenorientierte, künstliche, gegenständliche Gebilde definiert. Sie lassen sich verallgemeinert in einer Systembeschreibung darstellen (vgl. Abbildung 28).

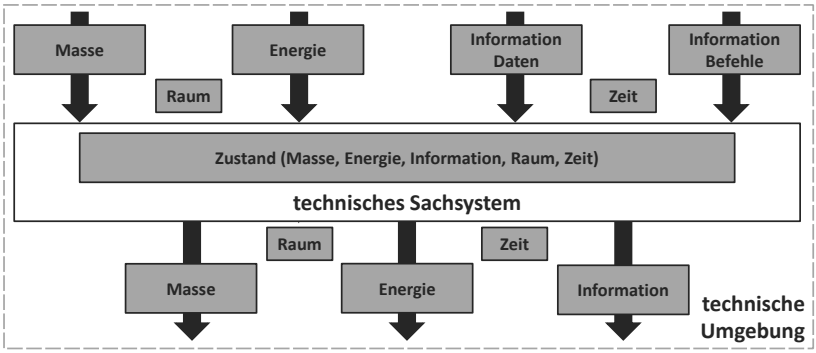

Abbildung 28: Systembeschreibung eines technischen Sachsystems nach Ropohl *(2009)*

Sachsysteme sind demnach von ihrer Umgebung durch eine Systemgrenze klar abgetrennte Objekte, welche, abhängig von Masse, Energie, Information, Raum und Zeit, unterschiedliche diskrete und nichtdiskrete Zustände annehmen können. Ein derartiges System kann durch entsprechende Eingangsgrößen – Masse, Energie, Information – beeinflusst werden und mit den Ausgangsgrößen – ebenfalls Masse, Energie und Information – unter den Rahmenbedingungen Raum und Zeit mit seiner Umgebung wechselwirken.

Technische Sachsysteme lassen sich zu größeren Systemen zusammenfassen (Sach-Supersysteme) oder ggf. in kleinere Systeme unterteilen (Sach-Subsysteme). Ein System kann ein Supersystem und mehrere Subsysteme besitzen. Als Referenz für die Einteilung lässt sich ein Hierarchiemodell technischer Systeme aufstellen, welche die Sachsysteme vom elementaren Werkstoff bis hin zu einem Anlagenverbund hierarchisch einordnet (vgl. Abbildung 29). Die Begrifflichkeiten dieser Einordnung sind dabei als Orientierungshilfsmittel und nicht als eindeutige Abgrenzung zu verstehen.

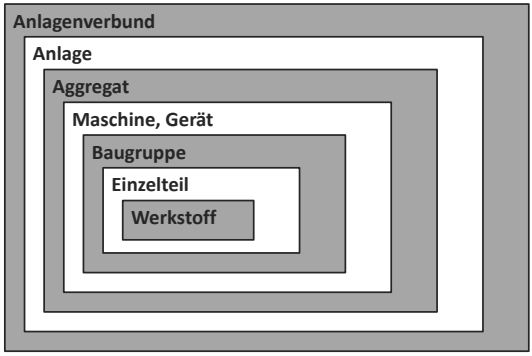

Abbildung 29: Hierarchie technischer Systeme in Anlehnung an Ropohl *(2009)*

Neben der hierarchischen Darstellung von Systemen und Systemverbindungen, wie eben beschrieben, ist zudem die strukturale und funktionale Sicht der Systeme zu berücksichtigen.

Strukturale Sicht – In der strukturalen Sicht wird das System als Ganzes mit miteinander verbundenen Subsystemen bzw. Elementen gesehen (vgl. Abbildung 30).

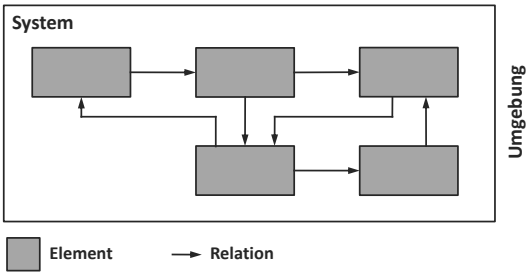

Abbildung 30: Strukturale Sicht auf technische Systeme nach Ropohl *(2009)*

Die strukturale Sicht ist vorwiegend nach innen gerichtet und betrachtet somit die Beschaffenheit der Komponenten und deren Integration in das Gesamtsystem. Die Existenz und Relationen der einzelnen Subsysteme und Komponenten untereinander führen zu einem unterschiedlichen Systemverhalten.

Funktionale Sicht – Dementgegen wird in der funktionalen Sicht das System als „Blackbox“ betrachtet, das über seine Eingänge und Zustände bestimmte Systemausgänge beeinflusst und so aus systemischer Gesamtsicht umgebungsrelevante Funktionen erfüllt (vgl. Abbildung 31). Es ist aus Systemsicht eine nach außen gerichtete Betrachtungsweise. Diese schafft eine technische Kapselung und damit Komplexitätsreduzierung, da nicht die Frage „Wie funktioniert das System?“, sondern „Was macht das System?“ gestellt wird.

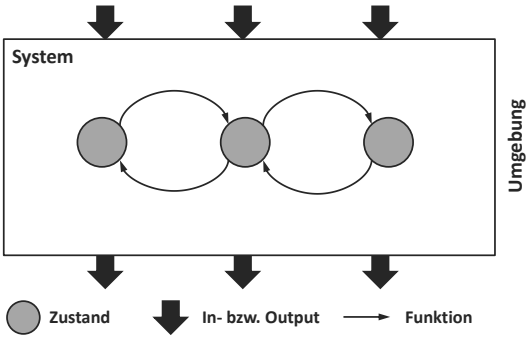

Abbildung 31: Funktionale Sicht auf technische Systeme in Anlehnung an Ropohl *(2009)*

5.3 Funktionsorientierte Modularisierung für Robotersysteme

5.3.1 Allgemeines

Überträgt man die allgemeine Systemtheorie auf den Themenkomplex der Robotersysteme und deren Konfiguration, wird deutlich, dass die Generierung von Systemgrenzen – die Modularisierung – auf unterschiedlichster Ebene erfolgen kann. Abbildung 32 zeigt das hierarchische Systemmodell angepasst auf ein Robotersystem.

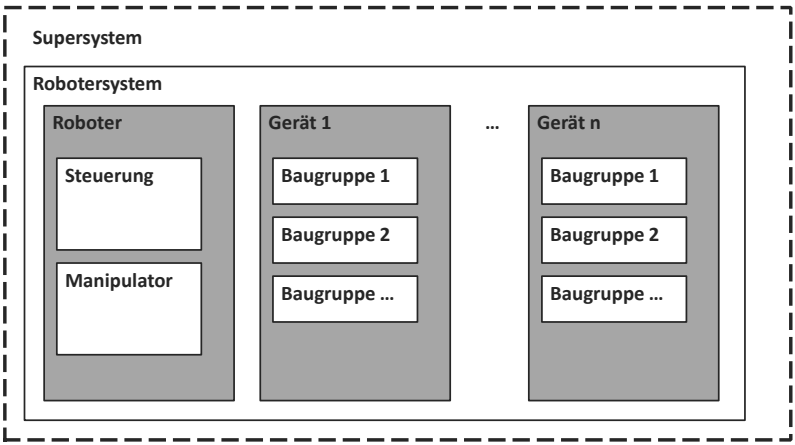

Abbildung 32: Hierarchisches Robotersystemmodell

Modularisierung erfordert nun in diesem Zusammenhang die einheitliche, sinnvolle Definition von Systemgrenzen sowohl in funktionaler (nach außen gerichtete), als auch in strukturaler (nach innen gerichtete) Sicht.

Funktionale Sicht

Die Module sollen aus Sicht des Roboters gewisse Funktionen erfüllen, welche die Bearbeitungsaufgabe vorantreiben, oder unterstützen (z. B. soll ein Greifer greifen und loslassen können). Abbildung 33 zeigt die funktionale „Blackbox“-Sicht der Module.

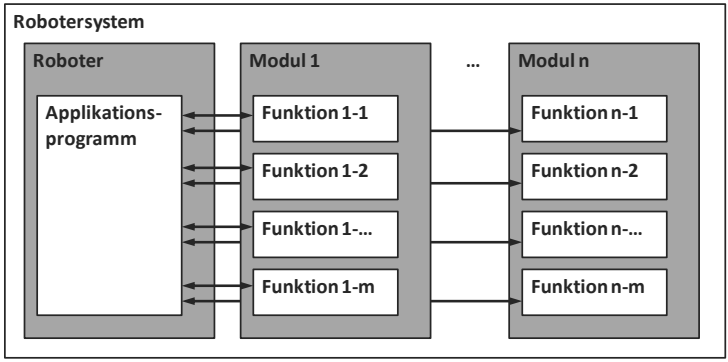

Abbildung 33: Funktionale, nach außen gerichtete Modulsicht

Dabei kann ein Modul eine, aber auch mehrere Funktionen erfüllen. Diese Betrachtung kapselt die Funktionsweise, reduziert damit die Komplexität und stellt die Funktion als abstrakte Eigenschaft des Geräts dar. Je nach Definition der Funktionen sind die Systemgrenzen anders zu ziehen. Diese Sichtweise soll anhand des Beispiels eines Druckluft-Greifers verdeutlicht werden: Ist die nach außen wirkende Funktion *greifer_oeffnen()*, ist eine andere Systemgrenze zu wählen als bei der Funktion *druckluft_schalten()*. Der Modulumfang ist abhängig von der gewünschten, nach außen gerichteten Funktionalität des Geräts.

Strukturale Sicht

Gleichzeitig muss aus der nach innen gerichteten, strukturalen Sicht das Systemverhalten so abgebildet werden, dass die geforderten Funktionen innerhalb der definierten Modulgrenzen erfüllt werden können (vgl. Abbildung 34). Das bedeutet, dass die Subsysteme des Moduls alleine in der Lage sind, die nach außen gerichteten Funktionen durchzuführen. Unter Modularisierung wird also in diesem Zusammenhang immer die ganzheitliche Betrachtung der funktionalen und strukturalen Sicht verstanden. Sind die Systemgrenzen über die verschiedenen Sichtweisen einheitlich definiert, so bedeutet dies, dass alle Bestandteile (Mechanik, Elektrik und Software) zur Ausführung einer bestimmten Funktion in einem Modul zusammengefasst und nicht über verschiedene Geräte in der Zelle verteilt sind. Das ist insofern konsequent, da bei dem Hinzufügen oder

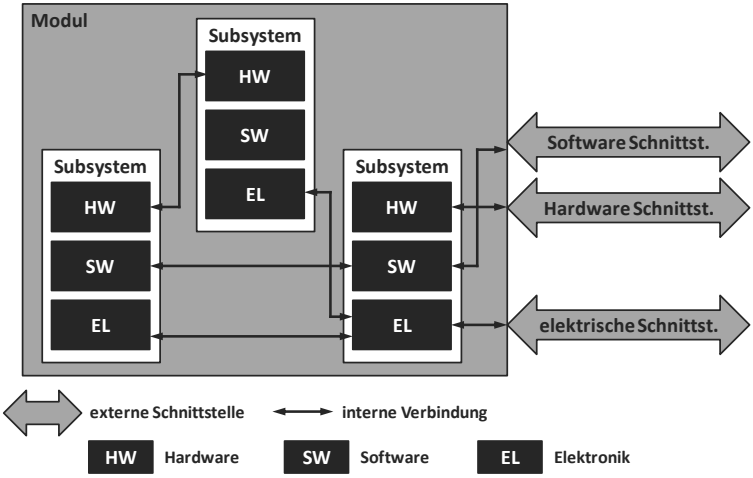

Abbildung 34: Strukturale, nach innen gerichtete Modulsicht

Entfernen eines Moduls keine elementaren Bestandteile zur Funktionsdurchführung fehlen bzw. nicht mehr benötigte Bestandteile das System negativ beeinflussen können.

5.3.2 Schnittstellenbetrachtung

Die Wechselwirkungen zwischen Systemen und Super- bzw. Subsystemen – hier zwischen Industrieroboter und Modulen – erfolgen ausschließlich über deren Schnittstellen. Schnittstellen sind also kritische Punkte in wandelbaren Produktionssystemen (Nyhuis et al. 2008). Auch die Schnittstelle zur Umgebung, z. B. zur Energieversorgung, spielt dabei eine wichtige Rolle. Gängige Schnittstellenarten, die in Anlehnung an das Systemmodell Masse, Energie und Informationen übertragen, sind in Hardware-Schnittstellen (z. B. Druckluftversorgung oder Kraftübertragung), elektrische Schnittstellen (z. B. zur Stromversorgung) und Software-Schnittstellen (z. B. zur Datenübertragung) untergliedert. Zur Einordnung der Schnittstellentypen wird hier lediglich die Primärfunktion herangezogen. Daher wird beispielsweise der Stecker einer elektrischen Schnittstelle nicht als mechanische Schnittstelle gesehen. Ein Modul kann mehrere Schnittstellen zu seiner Systemumgebung aufweisen. Für

eine möglichst hohe Kompatibilität wird generell eine Standardisierung der verschiedenen Schnittstellen als Lösung gesehen (Nyhuis et al. 2008).

Mechanische und elektrische Schnittstellen

Im Bereich der elektrischen Schnittstellen haben sich bereits Standards durchgesetzt. So gibt es feste Spannungs- und Leistungsgrößen mit normierten Steckern und Kennwerten (z. B. 230 V/ 16 A) (DIN 60038 2002, DIN 60320 2008). Elektrische Schnittstellen sind dabei nur selten zwischen Systemen mit funktionaler Verbindung, die den Produktionsvorgang betreffen, erforderlich. Die Verbindung besteht meist zu einem in der Umgebung angesiedelten Versorgungsnetz. Die Auslegung der Schnittstelle hinsichtlich Leistungsfähigkeit ist jedoch vom Anwendungsfall abhängig. Diese Schnittstellen sind in der Regel in Produktionsumgebungen vorhanden.

Mechanische Schnittstellen erfordern eine differenziertere Betrachtung. Sie sind hier in die Bereiche pneumatische Schnittstellen, mechanische Schnittstellen zwischen Modul und Umgebung und mechanische Schnittstellen zwischen Modul und Industrieroboter (Supersystem) gegliedert.

Für *pneumatische Schnittstellen* gilt Ähnliches wie für die elektrischen Schnittstellen. Anschlüsse, Drücke und Durchflussmengen sind großteils genormt (z. B. DIN 16030 (2005)) und lassen sich durch einfache Dimensionierung an die Einsatzsituation anpassen. Auch hier steht in Produktionsumgebungen meist eine entsprechende Infrastruktur zur Verfügung.

Mechanische Schnittstellen zwischen Modul und Umgebung sorgen für die Befestigung der Module in der Umgebung, bspw. die Befestigung einer Positionier- oder Spannvorrichtung auf dem Hallenboden. Diese Verbindungen sind sehr abhängig von der Situation und dem Einsatzfall. Daher sind ggf. Halterungen oder Befestigungen zu installieren, die z. B. mechanische Kräfte übertragen.

Mechanische Schnittstellen zwischen Modul und Industrieroboter bestehen nur dann, wenn das Modul vom Roboter geführt wird. Das Modul ist in diesem Fall am Flansch des Roboters befestigt und wird als Endeffektor bezeichnet. Diese spezielle mechanische Schnittstelle ist in den Normen DIN 11593 (1996) und DIN 9409 (2004) spezifiziert.

Mechanische und elektrische Schnittstellen, die innerhalb von Robotersystemen und zu der Umgebung bestehen, weisen bereits ein hohes Maß an Standardisierung auf oder sind so prozess- und situationsindividuell, dass eine Vereinheitlichung unter den gegebenen Rahmenbedingungen nur schwer möglich ist.

Informationstechnische Schnittstellen

Wie bisherige Entwicklungen von Bussystemen und Robotersteuerungen gezeigt haben, konnten sich einheitliche Standards bei informationstechnischen Schnittstellen nicht durchsetzen. Sowohl Roboterhersteller als auch Hersteller von Industrial-Ethernet-Geräten haben ihre eigenen Protokolle und Schnittstellen. Dies hat verschiedene Gründe:

Bisherige Versuche, Robotersprachen und Architekturen zu vereinheitlichen, scheiterten, da jeder Roboterhersteller seine eigenen Konzepte verfolgt. Es lässt sich bei Industrierobotern eine gemeinsame Basismenge an Funktionalitäten identifizieren. Darauf aufbauend bieten Roboterhersteller eigene Funktionen an, oft auf bestimmte Anwendungen zugeschnitten. (Keibel 2003)
Diese funktionalen Unterschiede stellen Unterscheidungsmerkmale zu Mitbewerbern dar. Daher ist eine Vereinheitlichung von Schnittstellen nicht wahrscheinlich.

Feldbusse sind je nach Einsatzzweck durch unterschiedliche Leistungsmerkmale gekennzeichnet. Zudem sind in einigen Branchen bestimmte Feldbusprotokolle etabliert. Dies führt soweit, dass sogar die internationale Norm DIN 61158 (2009) selbst verschiedene Feldbusse normiert. Eine Vereinheitlichung der Feldbusprotokolle ist daher nicht absehbar.

Hersteller von Geräten differenzieren sich häufig durch individuelle Funktionalitäten am Markt. Alleinstellungsmerkmale, gerade bei Nischenprodukten, sind Spezialeigenschaften und Funktionen der Geräte. Eine Standardisierung würde diesen Wettbewerbsvorteil nicht unterstützen.

Die informationstechnische Schnittstelle ist, bedingt durch die technischen und wirtschaftspolitischen Rahmenbedingungen auf der einen Seite und der hoch individuellen Gerätefunktionalität auf der anderen Seite, nicht eindeutig standardisierbar. Es existieren jedoch sowohl auf verschiedenen Ebenen als

auch in verschiedenen Branchen unterschiedliche Standards. Bei der Wahl der informationstechnischen Schnittstelle sind daher branchenspezifische Standards einzuhalten und Funktionalitäten vollständig abzubilden.

5.3.3 Einsatz von Geräteprofilen

In diesem Kontext ist auch der Einsatz von Geräteprofilen zu diskutieren. Geräte- oder auch Kommunikationsprofile sind ein Ansatz zur Kommunikationsstandardisierung im Bereich speicherprogrammierbarer Steuerungen in Verbindung mit produktionstechnischen Kommunikationssystemen. CAN-basierte Bussysteme bspw. weisen, induziert durch Geräteprofile, standardisierte Schnittstellen in dem sogenannten CAN-Application-Layer zu der Anwendung auf – Schicht 7 im ISO/OSI-Modell (Boterenbrood 2000). Zudem existiert mit dem DIN-Fachbericht 62390 (2009) ein Leitfaden für Geräteprofile in der Automatisierungstechnik, welcher ein Verständnis über deren Gestalt und Verwendung liefert.

„Profile definieren für eine Klasse von Geräten eine gemeinsame Menge von Funktionalitäten in einem vorgegebenen industriellen Gebiet, um so den Systementwicklern, den Systemintegratoren und dem Instandhaltungspersonal zu ermöglichen, mit Profil-basierten Geräten ohne eine spezielle Werkzeugkonfiguration umzugehen. Dies ermöglicht auch die konsistente Strukturierung und Semantik der Funktionalität der Geräte.“ (DIN-Fachbericht 62390 2009)

Geräteprofile sind damit ein herstellerunabhängiges, sehr wohl aber geräteabhängiges Pendant zu Gerätebeschreibungssprachen, die Steuerungshersteller in ihren Projektierungswerkzeugen verwenden. Auch in dem Gerätemodell des Leitfadens ist vornehmlich die informationstechnische Schnittstelle zu einem übergeordneten Steuerungssystem Betrachtungsgegenstand.

Bei der Verwendung von Geräte- und Kommunikationsprofilen werden verschiedene Kompatibilitätsebenen definiert (vgl. Abbildung 35). Diese bestimmen den Grad der Zusammenarbeit von Geräten, die Profile verwenden.

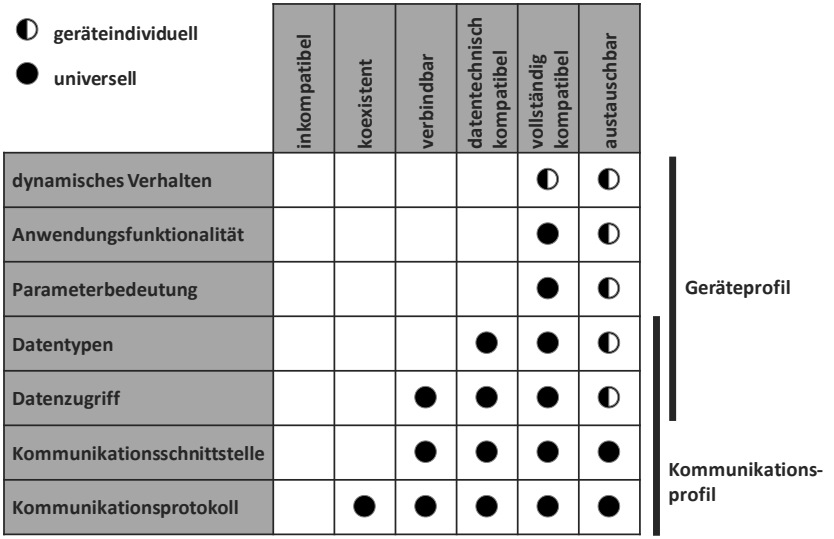

Abbildung 35: Kompatibilitätsebenen von Geräte- und Kommunikationsprofilen nach DIN-Fachbericht *62390 (2009)*

„Wenn ein Gerät vom Anwender programmierbar ist, können seine Merkmale“ … „(z. B. Parameter und Verhalten), nicht vollständig im Profil beschrieben werden. Profil-Verfasser können sich jedoch sowohl auf allgemeine gemeinsame Funktionen wie Start, Stopp und Rücksetzen einigen, als auch auf Kennzeichnung und Prozess-Ein/Ausgänge.“ (DIN-Fachbericht 62390 2009)

Eine Austauschbarkeit von Geräten ist also nur gegeben, wenn sie den identischen Funktionsumfang haben und darüber hinaus auch die (fast) identische Gerätebeschreibung aufweisen. Der Leitfaden bildet eine Basis für Arbeitsgruppen, einheitliche Produktklassenprofile zu entwickeln. Neue Produkte und Technologien erfordern erneut die Definition weiterer Profile und Profilklassen.

Geräteprofile bieten im Bereich der speicherprogrammierbaren Steuerungen die Möglichkeit, die Interoperabilität auf der Applikationsebene zu steigern. Sie sind jedoch wieder auf Standardisierung angewiesen. Herstellerspezifische Funktionen und Konventionen reduzieren den Nutzen von Profilen auf die unteren Schichten der Kommunikationsebene.

Bei dem Entwurf neuer Methoden sollten daher die Möglichkeiten und Vorteile von Geräteprofilen genutzt werden. Sie tragen in der heutigen Form (wegen der erforderlichen Standardisierung) allerdings im Bereich der Robotersysteme nicht zur automatischen Konfiguration bei.

5.3.4 Zusammenfassung

Die Modularisierung von Geräten zur Plug&Produce-Befähigung bedarf sowohl einer funktionalen als auch einer strukturalen Modulsicht (vgl. Abbildung 36).

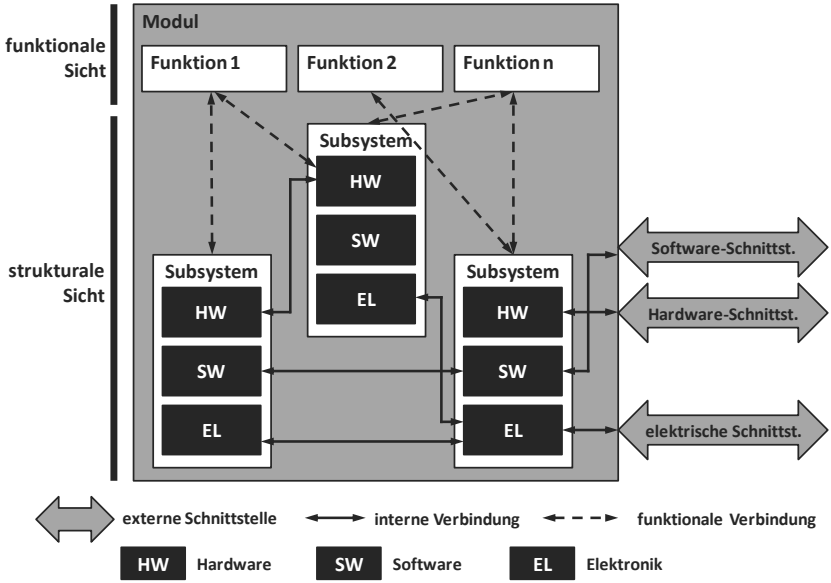

Abbildung 36: Schematische Modulstruktur

Schnittstellen definieren die Verbindung zu anderen Modulen. Während mechanische und elektrische Schnittstellen bereits weitestgehend standardisiert sind, ist bei den informationstechnischen Schnittstellen eine Standardisierung aufgrund der Diversität nicht möglich. Die Anwendung von Geräteprofilen ist nutzbringend, darf aber die Funktionalität der automatischen Konfiguration

nicht einschränken. Im Folgenden wird ein Vorgehen beschrieben, das eine Gestaltung von Geräten für den Plug&Produce-Einsatz ermöglicht.

5.4 Gestaltung funktionsorientierter Plug&Produce-Module

Aus den Erkenntnissen zum Aufbau bzw. zur Struktur von Modulen, bestehenden Geräteprofilen und Schnittstellen, kann ein Ablauf zur Gestaltung funktionaler Module für Plug&Produce-Robotersysteme abgeleitet werden. Dieser dient sowohl zur Bewertung bestehender Geräte als auch als Hilfsmittel zur Neugestaltung. In dem hier aufgezeigten Kontext sind die gerätetechnischen Voraussetzungen hinsichtlich Datenverarbeitungsfähigkeit zu berücksichtigen. Der Gestaltung funktionaler Module bzw. der Umgestaltung bestehender Geräte und Equipment zu entsprechenden Modulen kommt demnach eine große Bedeutung zu. Ein effizienter Weg zur Gestaltung solcher Module ist folgendes dreistufiges Vorgehen (vgl. Abbildung 37):

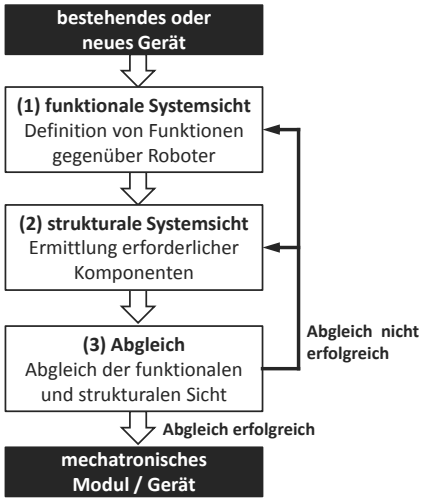

Abbildung 37: Dreistufiger Ablauf zur Modularisierung

(1) Funktionale Systemsicht – Identifizierung der Modulfunktionen; (2) Strukturale Systemsicht – Bestimmung der für die Ausführung der Funktionen

erforderlichen Bestandteile (Subsysteme, wie Hardware, Software, Elektrik); (3) Abgleich der Sichten – Vergleich der funktionalen bzw. strukturalen Sicht und Definition der Schnittstellen.

Am Beispiel des pneumatischen Greifers, der über ein Bussystem mit dem Roboter verbunden ist, soll das Prinzip des Modulaufbaus verdeutlicht werden. Die Modulgestaltung wäre wie folgt: Die Funktionen, die der Greifer erfüllen soll sind *greifen()* und *loslassen()*. Zur Ausführung dieser Funktionen werden zum einen die mechanischen Bestandteile (der Greifer selbst und der mechanische Teil des pneumatischen Ventils), zum anderen die elektrischen Bestandteile (hier der elektrische Teil des Ventils und die elektrischen Bestandteile der Buskomponente) benötigt. Der Softwarebestandteil ist in der Buskomponente angesiedelt. Diese setzt die Bussignale in ein Signal für das Ventil um. Schließlich wird die funktionale mit der strukturalen Sicht abgeglichen. Dazu wird überprüft, ob alle zur Funktionsdurchführung erforderlichen Komponenten innerhalb der Systemgrenzen liegen. Die mechanischen, elektrischen und softwareseitigen Schnittstellen werden definiert. Bei der Gestaltung der mechanischen und elektrischen Schnittstellen sollten nach Möglichkeit geltende Standards für das entsprechende Einsatzgebiet berücksichtigt werden.

6 Methode zur automatischen Konfiguration

6.1 Allgemeines

Aufbauend auf den Anforderungen und den Gestaltungsrichtlinien zur Modularisierung werden in diesem Kapitel kommunikations- und systemtechnische Vorüberlegungen zu Plug&Produce-Robotersystemen dargestellt. Daraus wird eine neuartige Methode zur automatischen Konfiguration abgeleitet. Es folgt die Darstellung des Wirkprinzips und der Funktionsweise. Darauf aufbauend wird die dafür erforderliche Informationsverarbeitung erläutert und schließlich ist der Ablauf der Methode beschrieben.

6.2 Vorüberlegungen im Kontext der automatischen Konfiguration

6.2.1 Informationsangebot und -nachfrage

Die Kommunikation zwischen Roboter und Peripheriegeräten ist in der Konfigurationsphase und der Betriebsphase getrennt voneinander zu betrachten. Während der Betriebsphase prägen die Rahmenbedingungen der industriellen Kommunikation den Informationsaustausch. Dies erfordert im Vorfeld eine genaue Definition der Echtzeitübertragung und der Verarbeitung von Prozessdaten bis in die funktionale Kommunikationsebene. Für die Festlegung dieser Eigenschaften in der Robotersteuerung und Programmierumgebung sind Informationen über die angeschlossenen Peripheriegeräte, wie bspw. deren Eigenschaften hinsichtlich ihres Kommunikationsverhaltens (vgl. Kapitel 4), erforderlich.
Während der Konfigurationsphase haben demnach die zu konfigurierenden Systeme Informationsbedarfe. Sie benötigen Informationen, um die nachfolgenden Schritte – Applikationsprogrammierung und Produktivbetrieb – zu ermöglichen. Die Gesamtheit dieser Bedarfe ergibt sich aus der Summe der Einzelbedarfe (vgl. Abbildung 38 links). Der hier definierte Betrachtungsraum der zu konfigurierenden Zielsysteme umfasst Robotersteuerungen, Programmierumgebungen und die Anlagendokumentation.

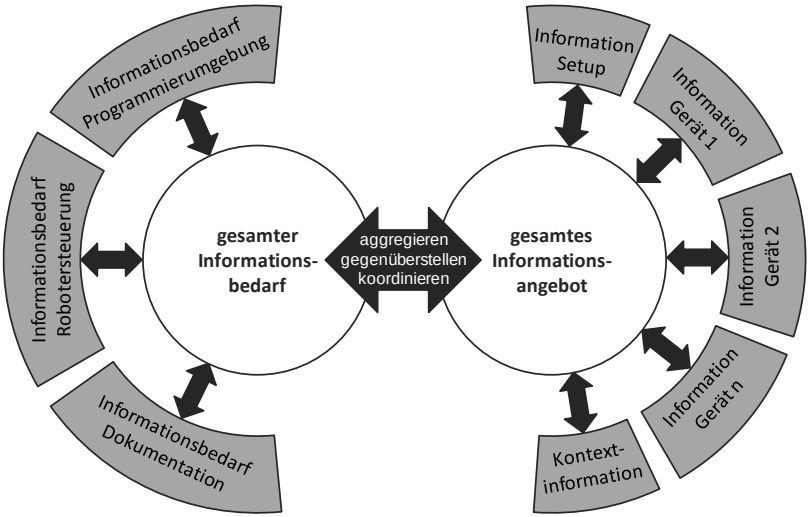

Abbildung 38: Gegenüberstellung des Informationsbedarfs und -angebots für die automatische Konfiguration

Zur Deckung des Informationsbedarfs muss auf der Seite der Peripheriegeräte ein Informationsangebot existieren. Für eine automatische Konfiguration können die Informationen entweder von den Geräten selbst in maschinenlesbarer Form bereitgestellt werden (z. B. Funktionen oder Prozessvariablen), sind abhängig vom Aufbau des Systems (z. B. geometrische Positionen), ergeben sich aus der Auflösung von systemischen Freiheitsgraden und Abhängigkeiten (z. B. Mapping – Variablenzuordnung) oder müssen im Fall von nicht erfassbaren Setup-spezifischen Informationen (z. B. Position eines Geräts) manuell bereitgestellt werden. Mögliche Informationsquellen sind in Abbildung 38 rechts dargestellt.

Der Informationsbedarf muss mit den Informationsangeboten weitestgehend vollständig gedeckt werden. Dazu sind die Informationsangebote zu aggregieren, die informatorischen Zusammenhänge bzw. Freiheitsgrade zu koordinieren und diese dem spezifischen Informationsbedarf gegenüberzustellen. In Abbildung 38 ist schematisch der Abgleich des Informationsangebots und des Informationsbedarfs dargestellt.

6.2.2 Einheitliche Kommunikationsbasis

Zur Übertragung der Informationen ist ein Kommunikationsnetz erforderlich. Grundsätzlich können beliebige Kommunikationssysteme dafür verwendet werden, welche es ermöglichen, die Geräteinformationen zu übertragen. Industrial-Ethernet-Netzwerke, die neben der Echtzeitkommunikation noch über weitere gängige Kommunikationskanäle (z. B. TCP/IP oder UDP/IP) verfügen, eignen sich besonders für die Übertragung der Geräteinformationen. Im weiteren Verlauf sind die Erkenntnisse und die Methode zur übersichtlicheren Darstellung anhand von Industrial-Ethernet-Netzwerken erläutert.
Für die weitere Betrachtung wird eine einheitliche Kommunikationsbasis vorausgesetzt, also die Verwendung eines einheitlichen Bussystems in einer Anlage. Mit sogenannten Gateways – auch Protokollumsetzer genannt – ist es zwar möglich, industrielle Netzwerke unterschiedlicher Standards miteinander zu verbinden. Sie erhöhen jedoch die Systemkomplexität und können die Übertragungsrate der Prozessdaten reduzieren (Wellenreuther & Zastrow 2009).

6.2.3 Datenformate und Informationsbereitstellung

Die Existenz unterschiedlicher Bussysteme, geräteindividueller Fähigkeiten, die Unterstützung von Transportprotokollen oder Formaten, diverse Betriebssysteme bzw. eingesetzte Controller auf Geräten und Robotern sowie bereits bestehende herstellerspezifische Standards ermöglichen keine einheitliche Informationsbereitstellung hinsichtlich Datenformat und Übertragungskanal auf den Peripheriegeräten. Das bedeutet: Der Kommunikationskanal, über den die Informationen der Geräte übermittelt werden, und die dazugehörigen Protokolle sowie das Datenformat und die Semantik folgen unterschiedlichen Konventionen. Informationsnetzwerke erfordern auf beiden Seiten der Kommunikation einheitliche Protokollstacks (Meyer 2002). So wird eine Übertragung und Interpretation erst möglich. Insofern muss bei der Datenerhebung den Protokollkonventionen des Geräts gefolgt werden. Abbildung 39 zeigt auf Basis des ISO-OSI-Modells und des Ethernet-basierten TCP/IP und FTP Protokollstacks die Datenerhebung eines Gerätes. Wenn eine der aufeinander aufbauenden Schichten nicht vom Gegenüber unterstützt wird, kann keine Datenübermittlung stattfinden.

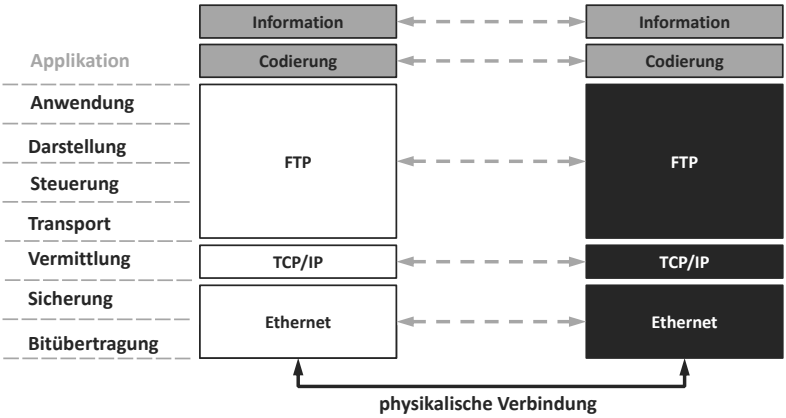

Abbildung 39: Kommunikationskanal mit einheitlichen Protokollen am Beispiel von TCP/IP und FTP

Ähnliches gilt für das Datenformat. Sind Syntax und Codierung der Daten bekannt, so können die Daten interpretiert, und damit die Informationen extrahiert werden. Ist dies nicht der Fall, ist eine Interpretation nicht möglich.

6.2.4 Schlussfolgerung

Für eine automatische Konfiguration sind die Informationen auf den Geräten maschinenlesbar zu hinterlegen und bereitzustellen. Verursacht durch die Heterogenität der Komponenten und Kommunikationssysteme verschiedener Hersteller sowie durch technologische Entwicklungen, sind die Definition und Einhaltung eines einheitlichen Formats sowie die einheitliche Bereitstellung dieser Informationen nicht praktikabel. So müssen die Informationen der Peripheriegeräte aus den gespeicherten Daten interpretiert, aggregiert und mit spezifischen Informationen, die sich aus dem Systemaufbau ergeben, angereichert werden. Die Informationen sind zielsystemspezifisch anzupassen und dort zu implementieren. Daraus resultiert auf der einen Seite ein Informationsbedarf und auf der anderen Seite ein Informationsangebot in unterschiedlichen Formaten und unterschiedlicher Semantik.

6.3 Methode (Plug&Produce)

6.3.1 Wirkprinzip der Methode

Ziele der Methode sind die Bereitstellung der Applikationsfunktionen sowie die Integration der spezifischen Geräteeigenschaften auf dem entsprechenden Zielsystem durch eine automatische Konfiguration. Sie ist so durchzuführen, dass die erforderlichen Kriterien – *Effizienz, Universalität und Einfachheit* (vgl. Abschnitt 4.3.1) – erfüllt werden. Damit soll, wie in Abbildung 40 dargestellt, nach der Konfiguration der Roboter in der Lage sein, die Funktionen der angeschlossenen Peripheriegeräte auszuführen bzw. deren Ausführung zu veranlassen.

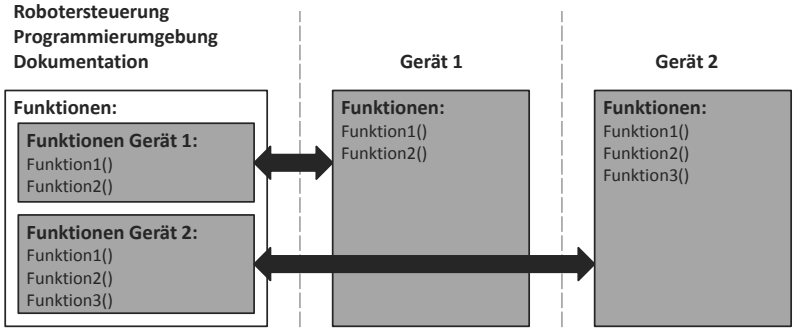

Abbildung 40: Wirkprinzip der Methode auf funktionaler Ebene

Die Methode zur automatischen Konfiguration soll die Kommunikation für den Anwender soweit abstrahieren und automatisieren, dass die unteren Kommunikationsschichten bis hin zur Applikationsschicht von dem Anwender ferngehalten werden. Abbildung 41 zeigt dies beispielhaft am System Robotersteuerung. Voraussetzung dafür sind nach Kapitel 5 erstellte oder erweiterte, funktionsorientierte modulare Geräte.

Dieses Ziel erfordert damit eine zielsystemspezifische Konfiguration mit jeweils systemindividuellen Maßnahmen. Eine Konfiguration der einzelnen Kommunikationsschichten ist bspw. bei dem Zielsystem Robotersteuerung erforderlich.

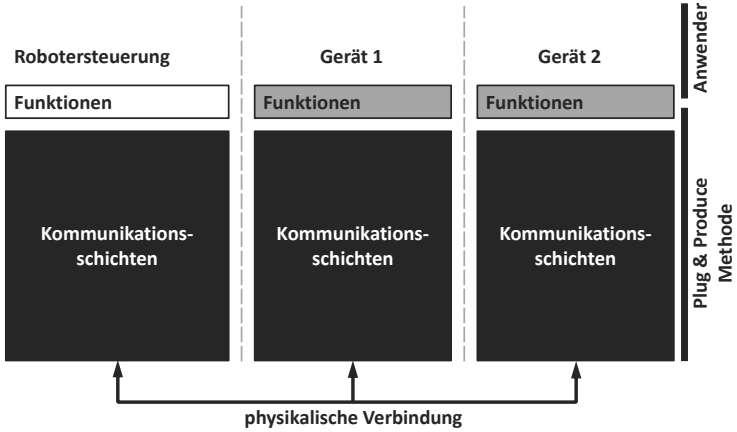

Abbildung 41: Abstraktion der Kommunikation bis hin zur Ebene der Gerätefunktionen

6.3.2 Aufbau der Plug&Produce-Methode

Die Methode nutzt das Informationsangebot der angeschlossenen Peripheriegeräte und erfüllt damit die Informationsnachfrage der Zielsysteme. Die Aggregation, die Gegenüberstellung und die Koordination dieser Informationen erfolgen mit einem sogenannten Zustandsmodell (vgl. Abbildung 42).

Dieses Zustandsmodell stellt eine abstrakte, standardisierte Beschreibung des Robotersystems dar und bildet damit einen der wesentlichen Bestandteile der Methode.

Neben dem Zustandsmodell unterstützen zwei weitere Datenbausteine die Methode: die Gerätebeschreibung und die Zielkonfiguration. Der Ablauf wird von einer Software durchgeführt, dem sogenannten Konfigurationsmanager, welche entweder separat an das Netzwerk angebunden oder auf der Robotersteuerung implementiert ist. Informationen über die einzelnen Peripheriegeräte und über den Roboter werden gesammelt und in dem Zustandsmodell gespeichert. Diese dienen anschließend dazu, für verschiedene Zielsysteme (die Robotersteuerung, die Programmierumgebungen und die Dokumentation) die Konfiguration zu erstellen und durchzuführen.

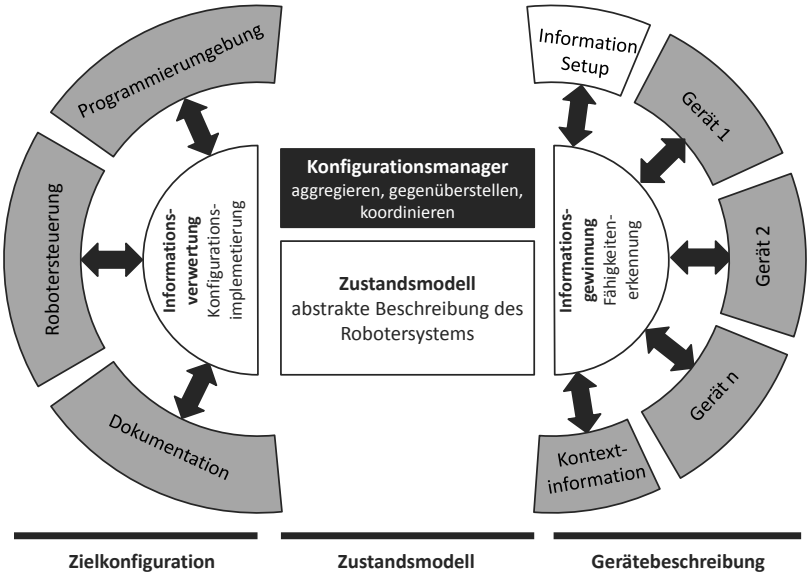

Abbildung 42: Aufbau und Elemente der Plug&Produce-Methode

Durch diesen Aufbau ist die Methode in der Lage, sowohl unterschiedlichste Geräte – mit deren ggf. individuellen Gerätebeschreibungen – als auch beliebige Robotersysteme und Programmierumgebungen zu unterstützen. Dieses Vorgehen benötigt als Datenformate eine Gerätebeschreibung, welche die erforderlichen, gerätespezifischen Informationen bereithält, ein Zustandsmodell und eine entsprechende Zielkonfiguration.

Zur weiteren Erläuterung ist zunächst die Informationsverarbeitung beschrieben. Hier werden die Datenhaltung und -übertragung behandelt. Die Datenmodelle und der Umgang mit der Format- und Übertragungsheterogenität sind beschrieben.

Anschließend folgt die detaillierte Beschreibung der sequenziellen Struktur und des Ablaufs der Methode. Diese Sichtweise beschreibt den Ablauf der Methode mit den einzelnen, dafür erforderlichen Schritten. Dabei werden Kommunikationsaspekte sowie die Kommunikationsschichten und deren Informationsimplementierung behandelt.

6.4 Informationsverarbeitung in der Methode

6.4.1 Konzept der Informationsverarbeitung

Bedingt durch die Heterogenität der Ziel- und Bussysteme sowie der Peripheriegeräte, ist neben der Informationsübertragung und -aufbereitung eine informationstechnische Unterstützung dieser Datenvielfalt durch die Methode erforderlich. Unter der Annahme, dass bei der Gerätebeschreibung und bei dem Zielsystem unterschiedliche Formate, keine einheitliche Semantik und mehrere Standards für den Übertragungsweg existieren, wurde für die Informationsübermittlung ein treiberbasiertes Konzept entworfen (vgl. Abbildung 43). Der Konfigurationsmanager übernimmt dabei das Handling der Treiber und koordiniert den Informationsaustausch.

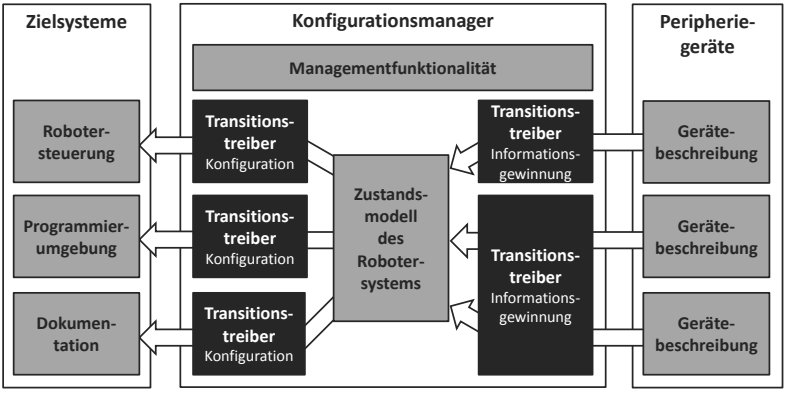

Abbildung 43: Informationsverarbeitung heterogener Datenformate und Übertragungswege während der Konfiguration

Die sogenannten *Transitionstreiber* beschreiben, über welche Protokolle die Daten von den Geräten zum Zustandsmodell und vom Zustandsmodell zu den Zielsystemen übertragen werden und wir sie zu interpretieren sind. Bemerkenswert ist hierbei, dass ein Treiber für unterschiedlichste Geräte und Zielsysteme verwendet werden kann, solange diese ihre Geräteinformationen oder Informationen über die Zielsystemkonfiguration nach den gleichen Konventionen bereitstellen. Diese Treiber können sich dabei an firmeninterne oder

Quasi-Standards anlehnen, sodass für die unterschiedlichsten Geräte (bspw. Greifer und Schweißgerät) derselbe Treiber verwendet werden kann.

Die Geräteinformationen (Gerätebeschreibung) werden, wie in Abbildung 43 dargestellt, mithilfe des Transitionstreibers zur Informationsgewinnung abgerufen, interpretiert und in das standardisierte Format des Zustandsmodells überführt. Dort werden die Informationen aller angeschlossenen Geräte systematisch abgelegt. Aus dem vereinheitlichten Modell lassen sich schließlich, mithilfe des *Transitionstreibers für die Konfiguration*, für das entsprechende Zielsystem aufbereitete Konfigurationsdaten ableiten und ggf. an dieses übertragen. Dafür werden die benötigten Informationen aus dem Zustandsmodell abgerufen und in das für das Zielsystem erforderliche Format übertragen. Der Treiber zeigt schließlich auf, wie die Informationen an das Zielsystem übermittelt werden.

Die drei Informationsbausteine, welche für dieses Konzept erforderlich sind, werden im Folgenden sowohl inhaltlich als auch vom informationstechnischen Aufbau näher betrachtet:

Die Zielsystemkonfiguration – Dies sind aufbereitete Informationen des Zustandsmodells, les- und interpretierbar für die zu konfigurierenden Systeme (Robotersteuerung, Programmierumgebung und Dokumentation), die Zielsysteme.

Das Zustandsmodell – Es repräsentiert ein standardisiertes, abstraktes Abbild der Roboterzelle in einer vereinheitlichten Beschreibung.

Die Gerätebeschreibung – Sie kann in unterschiedlichen (standardisierten) Formaten auf den Geräten hinterlegt und über unterschiedliche Kommunikationswege zugänglich gemacht werden.

Darüber hinaus ist die Architektur der *Transitionstreiber* beschrieben, welche die geräte- und kommunikationsindividuellen Daten in das einheitliche Datenformat des Zustandsmodells überführen und aus den Daten des Zielsystems die individuelle Konfiguration erstellen.

6.4.2 Zielsystemkonfiguration

Die drei verschiedenen Kategorien an Zielsystemen des Betrachtungsraums – Robotersteuerung, Programmierumgebung und Dokumentation – weisen jeweils unterschiedliche Informationsanforderungen auf. Die Erhebung dieser

Bedarfe erfolgte auf Basis der Dokumentation bestehender Systeme sowie der durchgeführten Informationsflussanalyse.

Konfiguration von Robotersteuerungen

Für die Konfiguration der Robotersteuerung sind folgende Informationen von Bedeutung: die Implementierung der verschiedenen Kommunikationsschichten für die Übertragung der Prozessdaten in Echtzeit während des Betriebs, das Mapping, also die Verknüpfung der Prozessdaten des Bussystems mit den Ein- bzw. Ausgängen des Roboters, die Beschreibung der Funktionen zur Erstellung des Applikationsprogramms und Informationen, die Einfluss auf die Bewegung und Regelung des Roboters haben, die kinematischen Parameter. Abbildung 44 zeigt die Informationsbausteine, welche für die Konfiguration einer Robotersteuerung erforderlich sind, eingeordnet in deren Protokollstack.

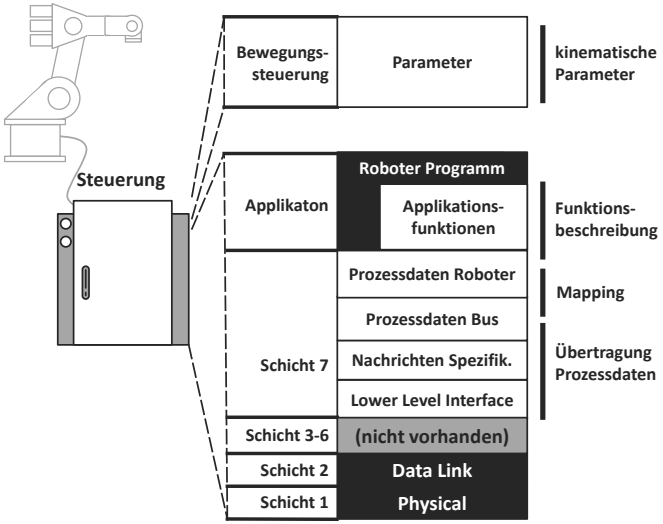

Abbildung 44: Informationsbedarf zur Konfiguration von Robotersteuerungen

1. *Übertragung Prozessdaten* – Die Informationsbedarfe der hardwarenahen Kommunikationsschichten zur Übertragung (vgl. Abbildung 44) sind stark abhängig von dem eingesetzten Bussystem. Dies ist durch spezifische Übertragungsprotokolle und -prinzipien sowie durch die verwendete

Hardware begründet. Für die meisten Bussysteme existieren Beschreibungssprachen (oft XML-basiert), welche die Definition aller für diese Schichten relevanten Informationen beinhalten. Beispiele hierfür sind: Powerlink – XML Device Description (XDD) bzw. Profibus/Profinet – Gerätestammdatendatei (GSD).

2. *Mapping* – Das Mapping definiert die informationstechnische Verbindung der Prozessdaten (die Prozessvariablen der einzelnen Geräte) des Bussystems mit den Ein-/Ausgängen (z. B. digital bzw. analog) des Roboters (vgl. Abbildung 45). Dafür sind Informationen erforderlich, welche die logische Verbindung der Prozessdaten mit den Eingängen und Ausgängen des Industrieroboters herstellen. Des Weiteren werden Informationen benötigt, welche die Überführung aller verfügbaren Datenformate der Prozessdaten des Bussystems in das Format der Ein- und Ausgänge des Robotersystems beschreiben (bspw. Bitwertigkeit, Größe oder Vorzeichen).

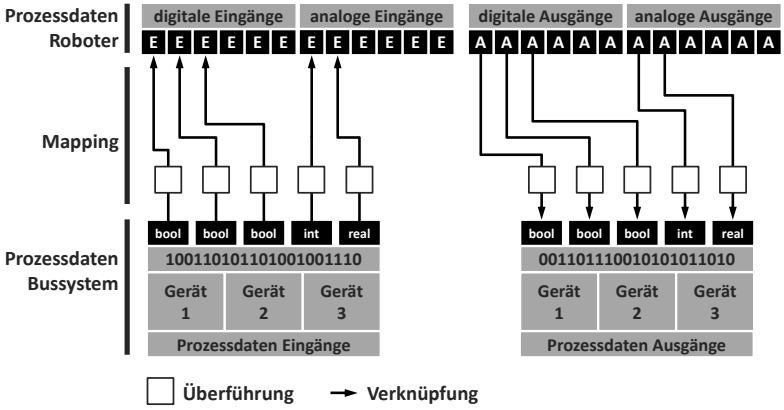

Abbildung 45: Schematische Darstellung des Mappings

3. *Funktionsbeschreibung* – Die Funktionsbeschreibung besteht aus den diversen Applikationsfunktionen, die zur Ansteuerung der Geräte erforderlich sind. Sie beschreiben die Logik der Kommunikationssequenz (z. B. Handshake oder Fehlerbehandlung) zwischen Robotersteuerung und Peripheriegeräten. Diese Funktionen nutzen dafür die Ein- und Ausgän-

ge der Robotersteuerung. Die Steuerungen der verschiedenen Hersteller verwenden dafür eigene Programmiersprachen (z. B. KRL – Kuka Robot Language, vgl. Abbildung 46). Die Funktionen beinhalten neben dem Ablauf zudem eine Bezeichnung – in dem Beispiel *GreiferOeffnen()*.

```
DEFFCT GreiferOeffnen()

;Setze Ausgang zum Schließen des Greifers
$OUT[5] = True

;Warte auf Sensor, dass Greifer nicht mehr offen
waitfor $IN[4] = False

;Warte auf Sensor, dass Greifer geschlossen
waitfor $IN[3] = True

;Ende
END
```

Abbildung 46: Einfaches Funktionsbeispiel in der Kuka-eigenen Programmiersprache KRL

4. *Kinematische Parameter* – Sowohl statische als auch dynamische Parameter beeinflussen die Bewegung des Roboters. Statische Parameter sind diejenigen, welche die Koordinatentransformation bzw. -rücktransformation festlegen. Dazu zählen ebenso diverse Referenzkoordinatensysteme, wie die Werkzeug-, Werkstück-, Basis- oder Weltkoordinatensysteme. Dynamische Parameter beziehen sich auf die zu bewegenden Massen, deren Schwerpunkte und die zugehörigen Massenträgheitsmomente. Diese bestimmen Bewegungs-, Beschleunigungs- und bspw. Durchbiegeverhalten des Roboters. Referenzkoordinatensysteme, Werkzeuge und damit die am Endeffektor wirkenden Massen sind während des Betriebs veränderlich (bspw. durch Werkzeugwechselsysteme). Diese Parameter sind daher gegebenenfalls mehrfach vorhanden, sodass bei Bedarf zwischen diesen umgeschaltet werden kann. Zur Erhöhung der Transparenz und zur Vereinfachung der Fehlerbehebung sind den Datensätzen jeweils Namen zugeordnet.

Konfiguration von simulationsbasierten Programmierumgebungen

Die simulationsgestützte Offlineprogrammierung dient der Erstellung von Roboterprogrammen in einer 3D-Simulation. Diese zeichnen sich durch drei we-

sentliche Informationsbedarfe aus (vgl. Abbildung 47), welche entlang des Programmierprozesses der Offlineprogrammierung – Aufbau des Zellenlayouts, Erstellung der Roboterbewegung und Steueranweisungen, Erzeugung des roboterspezifischen Programmcodes durch Postprozessoren – benötigt werden: die 3D-geometrischen und kinematischen Informationen, die logischen Informationen und allgemeine Informationen zu dem verwendeten Roboter und der Steuerung.

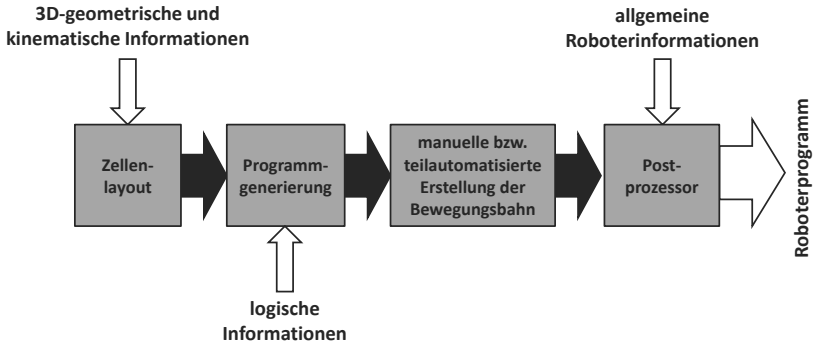

Abbildung 47: Informationsbedarf simulationsbasierter Programmierumgebungen

1. *3D-geometrische und kinematische Informationen* – Diese unterstützen die Erstellung der Bewegungsanweisungen im zu generierenden Roboterprogramm. Die Informationen beziehen sich einerseits, wie bei der Steuerung auch, auf die statischen, dynamischen und kinematischen Informationen sowohl des Roboters als auch der verwendeten Peripheriegeräte. Zur Darstellung, Kollisions- und Bahnberechnung, etc. sind andererseits geometrische 3D-Informationen sowie das Zellenlayout erforderlich.

2. *Logische Informationen* – Sie dienen der Erzeugung der Logik- bzw. Steueranweisungen für das Applikationsprogramm. Während der Programmierung werden Funktionen und Ein-/Ausgangswerte benötigt, welche es ermöglichen, Logik- und Steueranweisungen zu generieren, die während des Betriebs die gewünschten Funktionen ausführen. Diese Informationen entsprechen der Funktionsbeschreibung und dem roboterbezogenen Teil des Mappings.

3. *Allgemeine Informationen* – Darüber hinaus werden Informationen benötigt, die eine Aussage darüber geben, welcher Postprozessor für die Erzeugung des Roboterprogramms eingesetzt werden muss. Dies sind Herstellerinformationen sowie Informationen über Hard- und Softwarestand.

Konfiguration der Dokumentation

Technische Dokumentationen bestehen nach DIN 61355 (2009) aus der Gesamtheit an Dokumenten, die technische Erzeugnisse in deren Lebenszyklus beschreiben. Die dokumentationsrelevanten Informationen, welche durch die Systemkonfiguration induziert werden, stehen hier im Fokus, stellen aber nur einen Teil der gesamten Anlagendokumentation dar.

Dieser Teil der Dokumentation dient zwei maßgeblichen Zwecken: der Hilfestellung bei der Applikationsprogrammierung und Inbetriebnahme, also der Betriebsanleitung sowie der schriftlichen Fixierung der Struktur und des Aufbaus des Robotersystems – wie Konstruktionszeichnungen, Stücklisten, etc.

1. *Betriebsanleitung* – Die Hilfestellung bei der Inbetriebnahme besteht aus gerätespezifischen Informationen, die der Gerätehersteller dem Systemintegrator, Programmierer oder Instandhalter zur Verfügung stellt. Beispiele dafür sind: Zustandsautomat, Funktionsbeschreibung, Fehlerzustände und -meldungen. Hinweise zu Inhalt, Struktur und Aufbau von Anleitungen liefert die Norm DIN 62079 (2011).

2. *Dokumentation des Aufbaus und der Struktur der Anlage* – Im Fall von modularisierten Anlagen spiegelt sich die Modulstruktur in ähnlicher Form in der Dokumentation wieder. Ist also die Struktur der Anlage bekannt, so lässt sich daraus in der Regel die Struktur der Dokumentation ableiten. (Freisler 2008)
   Die Maschinenrichtlinie (2006) gibt die gesetzlich vorgeschriebenen Inhalte der Dokumentation vor. Die automatische Konfiguration kann dabei einen Beitrag zur Dokumentationserstellung leisten, indem die erforderlichen Informationen über Module und Anlagenstruktur bereitgestellt werden.

Zusammenfassung

Die Summe der Informationsbedarfe ist maßgeblich für die Gestaltung des Zustandsmodells. Tabelle 3 unterteilt die Bedarfe der Zielsysteme in die sechs Bereiche der Informationsflussanalyse und ordnet sie diesen zu:

1. *Hilfe* – Informationen über die verwendeten Geräte, bereitgestellte Funktionen, etc., die es dem Roboterprogrammierer erleichtern, das Applikationsprogramm zu erstellen

2. *Allgemeine Informationen* – Informationen über Hersteller, verwendete Bussysteme, etc.

3. *Geometriebeschreibung* – Beschreibungen von geometrischen Abmaßen, Gewichten und Massenträgheiten, Kinematiken, Informationen über Achsen und für die Bewegung relevanten Koordinatensystemen

4. *Funktionsbeschreibung* – Abbild von Funktionalitäten der angeschlossenen Geräte

5. *Prozessdaten Mapping* – Prozessdatenweiterleitung von Bussystem an Roboter und umgekehrt

6. *Echtzeitkommunikation* – Spezifikation der Echtzeitkommunikation (Zykluszeit, etc.) abhängig vom verwendeten Bussystem

6.4.3 Zustandsmodell

Zentrales Element der Methode ist das Zustandsmodell, welches ein standardisiertes modellhaftes Abbild des Robotersystems darstellt. Das Modell bildet die aktuelle Beschaffenheit dieser Systeme hinsichtlich der für die Konfiguration der Zielsysteme relevanten Parameter und Einstellungen in einer semantisch und syntaktisch vereinheitlichten Form ab. Dieses Modell fungiert damit als abstraktes, universelles Austauschformat und stellt somit ein Metamodell der Konfigurationsdaten dar.

● wird vollständig benötigt
◐ wird teilweise benötigt
- wird nicht benötigt

Informationsbereiche	Zielsysteme: Roboter-steuerung	simulative Programmier-umgebung	Dokumentation
Hilfe	-	-	●
Allgemeine Informationen	-	◐	●
Geometriebeschreibung	◐	●	◐
Funktionsbeschreibung	●	●	◐
Prozessdaten Mapping	●	◐	◐
Echtzeit Kommunikation	●	-	-

Tabelle 3: Informationsbereiche mit dem Informationsbedarf der Zielsysteme

Informatorische, semantische und systemische Modell-Definitionen

Zum einen ist eine inhaltlich informatorische Definition erforderlich. Zum anderen bedarf das Modell einer semantischen und systemischen Festlegung, welche es ermöglichen, die erforderlichen Informationen abzubilden.

Informatorische Modell-Definitionen – Grundsätzlich soll das Zustandsmodell den aktuellen Zustand des Robotersystems mit den Eigenschaften abbilden, welche für die Konfiguration erforderlich sind. Die inhaltliche Definition orientiert sich demnach an den Informationsbedarfen der Zielsysteme. Es werden die sechs bereits definierten Informationsbereiche als Grundlage herangezogen. Sollten, durch bspw. Weiterentwicklung der Robotersysteme oder neue Programmiermethoden spezifischer Robotersysteme, bzw. weitere neue Informationsbedarfe entstehen, so müssen diese über manuelle Eingabe während des Konfigurationsvorgangs zusätzlich erfasst werden (vgl. Abbildung 48). Ggf. wird eine Erweiterung des Zustandsmodells erforderlich.

Semantische Modell-Definition – Das Zustandsmodell muss die Modellierung von Semantik zulassen. Es muss ein einheitliches, aber erweiterbares Format aufweisen. Nur damit ist eine semantische Interpretation der in dem Modell gespeicherten Daten beliebiger Peripheriegeräte möglich.

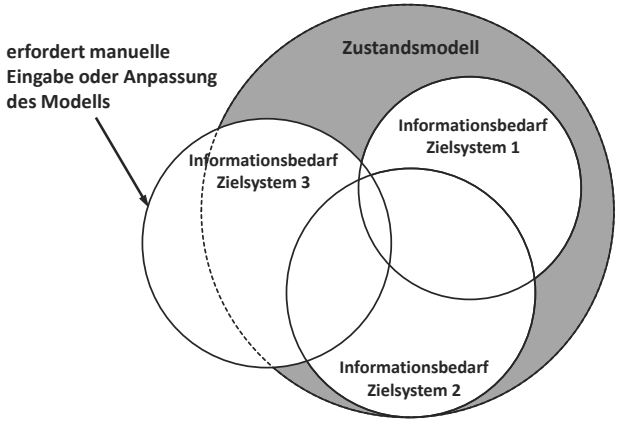

Abbildung 48: Informationsabdeckung des Zustandsmodells

Systemische Modell-Definition – Das Zustandsmodell dient ausschließlich der automatischen Informationsverarbeitung. Dieses Modell muss daher vollständig maschinenlesbar und -veränderbar sein. Um eine formale Richtigkeit und auch die informationstechnische Verarbeitbarkeit sicherzustellen, muss das Zustandsmodell auf seine syntaktische und semantische Korrektheit überprüft werden können.

Analyse bestehender Beschreibungssprachen

Grundsätzlich ist die technische Ausgestaltung des Zustandsmodells hinsichtlich Format und semantischer Darstellung für die Funktionsweise der Methode nicht maßgeblich. Die Nutzung bestehender, etablierter Beschreibungsformen ist jedoch insofern sinnvoll, als bereits die formale Korrektheit und Vollständigkeit sichergestellt sind. Im Folgenden werden Beschreibungsformen für das Modell auf Basis der Modell-Definitionen ausgewählt.

Im Umfeld der Automatisierungstechnik finden sich Beschreibungssprachen und -formen in verschiedenen Ausprägungen. Grundsätzlich sind alle grafisch orientierten Beschreibungssprachen, die nicht auf einer syntaktisch formellen bzw. maschinenlesbaren Beschreibung aufbauen, nicht für die modellhafte Beschreibung geeignet, da das Zustandsmodell zur automatischen Verarbeitung genutzt wird und nicht als optisches Hilfsmittel für den Menschen dient. Bekannte

Beispiele für diese grafischen Beschreibungsformen sind UML, SysML und Petrinetze. Diese grafischen Beschreibungssprachen beinhalten zudem meist Informationen über übergeordnete (planerische) Zusammenhänge und bilden keine konkreten Informationen zu den betreffenden technischen Aspekten ab. Austauschformate, zur Nutzung von Planungs- und Projektierungsdaten in verschiedenen computerbasierten Werkzeugen, eignen sich hingegen, da sie meist den erforderlichen Detaillierungsgrad erreichen und bereits über eine festgeschriebene Syntax und semantische Darstellung verfügen. Die bestehenden Beschreibungssprachen, wie bspw. PLCopen, CAEX, XDD oder VRML, decken nur jeweils einen Teil der erforderlichen Informationen ab. Für das Zustandsmodell ist demnach eine Kombination aus bestehenden Beschreibungsformen erforderlich.

AutomationML bietet eine Struktur für die Kombination verschiedener Beschreibungssprachen, welche unterschiedliche etablierte Datenformate unterstützt. Sie lässt eine Erweiterung der Beschreibungssprache um zusätzliche Formate zu. Die Festlegung dieser sowie die Fähigkeit diese interpretieren und austauschen zu können, obliegt den jeweiligen Anwendungen, die AutomationML verwenden. Bei der Entwicklung von AutomationML stand die Austauschbarkeit von Planungsdaten mit unterschiedlichen computergestützten Planungswerkzeugen im Vordergrund, um damit die Kosten für die Planung von Automatisierungslösungen zu reduzieren. Maßgebliches Ziel von AutomationML ist die möglichst vollständige Beschreibung kompletter Anlagen mitsamt den enthaltenen Komponenten. Der Mehrwert besteht vor allem darin, existierende Standards zu kombinieren und zu verknüpfen, um dadurch weitestgehend die Speicherung redundanter Informationen zu vermeiden. Kernfunktion ist die Referenzierung vorhandener Formate und die korrekte Interpretation der abgelegten Informationen zu den Anlagenkomponenten. Abbildung 49 zeigt die Grundarchitektur von AutomationML mit der Möglichkeit zur Erweiterung. Das Dachformat zur Beschreibung der hierarchischen Anlagentopologie bildet CAEX (Computer Aided Engineering Exchange). Das Format ist in der Norm IEC 62424 standardisiert. CAEX, eine XML-basierte Beschreibung, ist in der Lage, die Struktur einer Anlage in Form einer Objekthierarchie abzubilden. Der CAEX-Standard umfasst Aufbauregeln, die in einem XML-Schema definiert sind. Durch dieses Schema kann die syntaktische und strukturelle Richtigkeit von CAEX-Daten und damit auch AutomationML-Daten überprüft und sichergestellt werden. (Draht 2010; Mayr & Draht 2007)

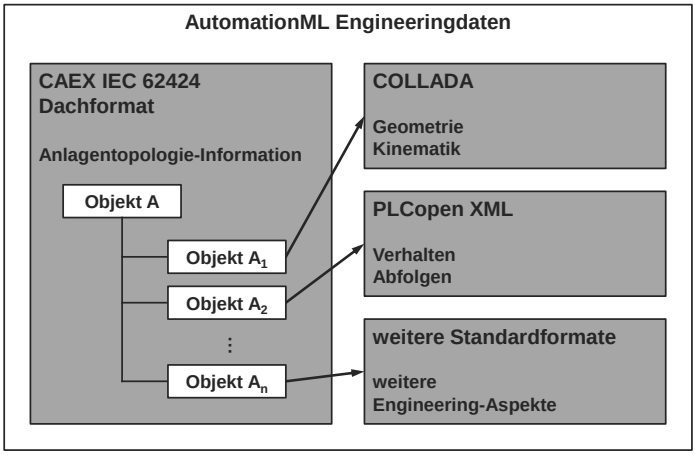

Abbildung 49: Architektur von AutomationML nach Draht *(2010)*

Nach Draht (2010) stellt CAEX mehrere Gestaltungsprinzipien bereit: Mit *Klassen und Bibliotheken* können wiederverwendbare Vorlagen vererbt, instanziiert und verfeinert werden. Eine *Instanz-Hierarchie* lässt die Beschreibung einer konkreten Anlage bzw. Teilanlage zu. Darin inbegriffen sind Objekte, Attribute, Schnittstellen und Relationen. *Schnittstellen, Relationen und Referenzen* ermöglichen eine Beschreibung von Verbindungen zwischen CAEX-Objekten und zu externen Dokumenten, wie anderen Beschreibungsformen. AutomationML stellt standardmäßig Verbindungen zu geometrischen Beschreibungen mit COLLADA™ (Collaborative Design Activity) (Barnes & Levy-Finch 2008) und zu Beschreibungen von Abläufen mit PLCopen her. Ein *Rollenkonzept* dient der semantischen Identifikation von Objekten. So können bspw. Informationen je nach der zugewiesenen Rolle unterschiedlich gefiltert und dargestellt werden.

Aufbau und Gestaltung des Zustandsmodells

In der hier beschriebenen Methode zur automatischen Konfiguration von Robotersystemen wird daher der Beschreibungsstandard AutomationML eingesetzt. Um AutomationML für diesen Einsatz zu qualifizieren, wird die Architektur im Sinne der Spezifikation angepasst und erweitert. Die Architektur wurde

dazu um weitere Formate ergänzt und die Attribute entsprechend den Informationsbedarfen angepasst. Daraus ergibt sich die in Abbildung 50 dargestellte, AutomationML-konforme Struktur des Zustandsmodells mit den jeweiligen Formaten.

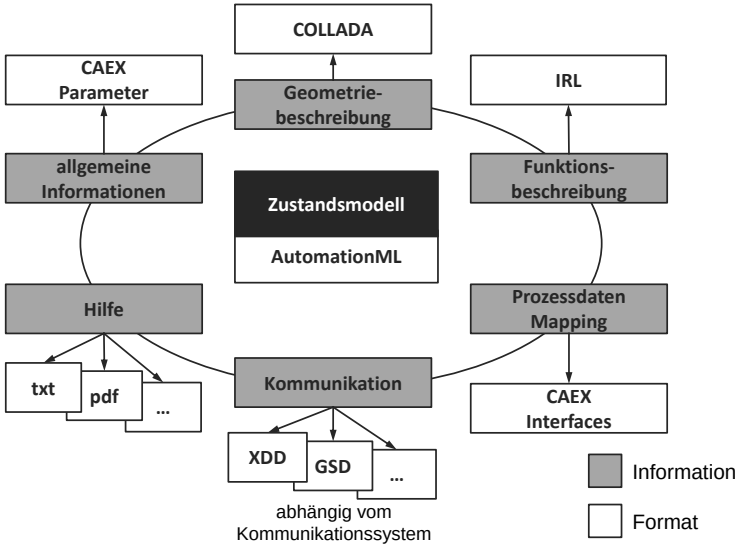

Abbildung 50: Informatorische Architektur des Zustandsmodells

Im Detail werden folgende Beschreibungsformen für die sechs Informationsbereiche des Zustandsmodells vorgeschlagen:

- *Hilfe* – Für Informationen der Hilfe werden gängige Beschreibungsstandards herangezogen, welche aus dem Office-Bereich bekannt sind (z. B. pdf, docx, txt, etc.). Für jedes Peripheriegerät wird im Zustandsmodell eine Beschreibung hinterlegt, die alternativ in einem der unterstützten Formate beschrieben ist.
- *Allgemeine Informationen* – Die allgemeinen Informationen sind in Form von Parametern im CAEX-Standard beschrieben.
- *Geometriebeschreibung* – Geometriebeschreibungen sowie kinematische Eigenschaften sind in dem AutomationML-nativen Datenformat COLLADA™ hinterlegt.

- *Funktionsbeschreibung* – Als Beschreibungsformat für die Gerätefunktionen wurde die herstellerneutrale Programmiersprache für Industrieroboter IRL (Industrial Robot Language) nach DIN 66312 (1996) gewählt. Trotz ihrer äußerst geringen Verbreitung – die Norm wurde aufgrund des geringen Interesses vonseiten der Roboterhersteller zurückgezogen – bildet IRL die Informationen zur funktionalen Beschreibung vollständig ab und ist damit zur Beschreibung im Zustandsmodell geeignet.

- *Prozessdaten Mapping* – Für das Mapping wird das beschriebene Schnittstellenkonzept von CAEX genutzt. Damit ist es möglich, die Ein- und Ausgänge des Roboters mit denen der Peripheriegeräte zu verbinden. So entsteht eine logische Verbindung der auszutauschenden Prozessdaten.

- *Echtzeitkommunikation* – Für Informationen über die *Kommunikation* werden die dort jeweils geltenden Beschreibungssprachen angewendet (z. B. Powerlink – XDD-Beschreibung oder ProfinetIO – GSD-Beschreibung). Damit sind die Bus-spezifische Echtzeitkommunikation und deren Prozessdatenbeschreibung abgedeckt.

Die Struktur des Zustandsmodells sowie der detaillierte Aufbau mit den entsprechenden Attributen ist in Anhang A abgebildet.

Funktionsweise des Zustandsmodells

Wird das Zustandsmodell für die automatische Konfiguration angewendet, kann dieses mit den Informationen der angeschlossenen Geräte gefüllt werden. Nach der Interpretation der Geräteinformationen durch den Transitionstreiber bilden diese zu den jeweiligen Informationsbereichen (Funktionsbeschreibung, etc.) neue Instanzen in den entsprechenden Formaten (vgl. Abbildung 51).

Interdependenzen zwischen den einzelnen Informationen sowie informationstechnische Freiheitsgrade (z. B. bei der Zuweisung freier Ein-/Ausgänge beim Mapping) sowie Kontextinformationen werden bei der Befüllung des Zustandsmodells aufgelöst und daher nicht abgebildet. Bei der Ableitung der Zielkonfiguration werden die Informationsbereiche aggregiert und über den Transitionstreiber für die Zielsystemkonfiguration zur Verfügung gestellt.

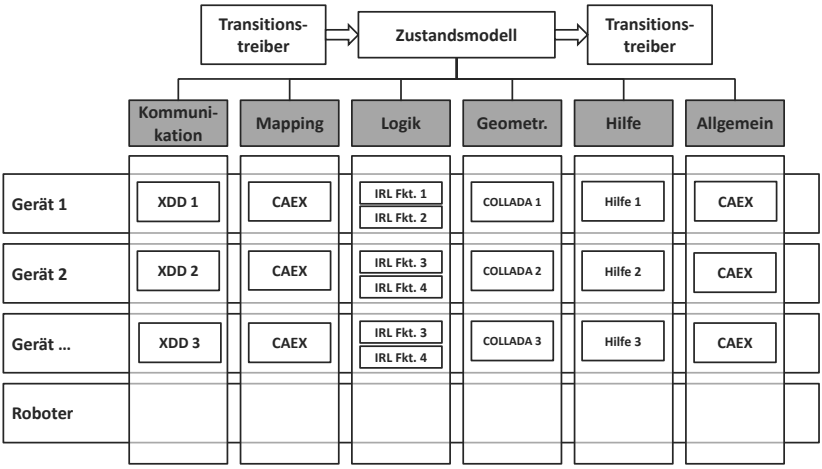

Abbildung 51: Funktionsweise des Zustandsmodell

6.4.4 Gerätebeschreibung

Zur Befähigung der Peripheriegeräte für eine automatische Konfiguration, wie in der oben genannten Methode gezeigt, müssen diese mit Geräteinformationen angereichert sein. Dies beinhaltet einerseits die Speicherung der Informationen und andererseits die Fähigkeit, die Informationen über das Netzwerk bereitzustellen (vgl. Abbildung 52). (Reinhart et al. 2011)

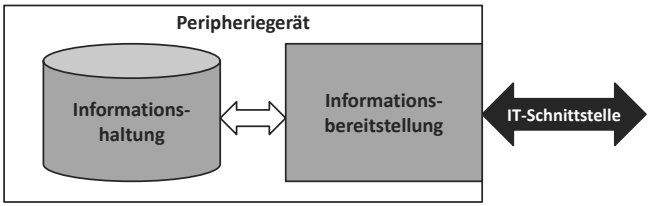

Abbildung 52: Gerätefähigkeiten zur Informationshaltung und -bereitstellung

Die Methode lässt durch die Verwendung der Transitionstreiber explizit zu, dass die Gerätebeschreibungen in unterschiedlichen Formaten bzw. Semantiken

hinterlegt werden. Die Informationsübertragung kann dabei über verschiedene Übertragungswege erfolgen. Eine Definition der Gerätebeschreibung ist damit nicht erforderlich. Inhaltlich sollen die Geräte die Informationen der sechs Informationsbereiche bereitstellen.

Anhang B zeigt einen Vorschlag zur Gerätebeschreibung am Beispiel eines Dreh-Kipp-Positionierers, welcher über Ethernet-Powerlink kommuniziert. Hier wurde die bestehende Powerlink-Gerätebeschreibung um die erforderlichen Informationen erweitert.

Der Übertragungsweg der Informationen an den Konfigurationsmanager kann über unterschiedliche Protokolle bzw. sogar über unterschiedliche Übertragungsmedien erfolgen. Abbildung 53 zeigt beispielhaft zwei unterschiedliche Protokollstacks von Peripheriegeräten: Die Informationen können über den im Industrial-Ethernet implementierten HTTP/IP-Stack (links) oder über die Managementfunktionalität des Bussystems (rechts) übertragen werden.

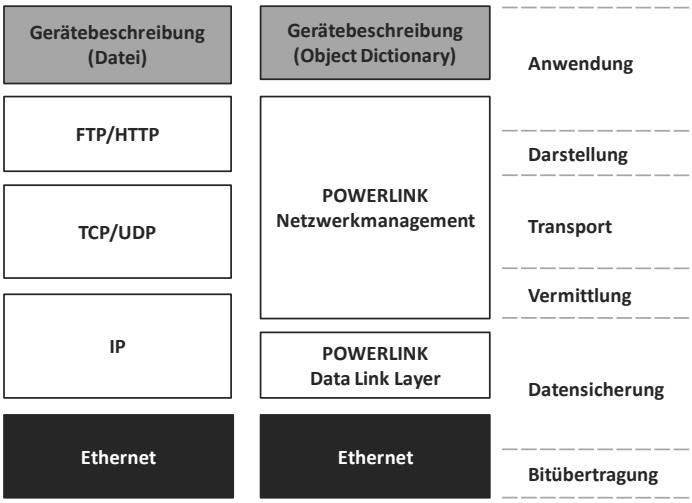

Abbildung 53: Informationsbereitstellung über das HTTP-Protokoll (links) und über Powerlink-Netzwerkmanagement-Funktionalität (rechts)

6.4.5 Transitionstreiber

Der Transitionstreiber hat zwei Aufgaben (vgl. Abbildung 54): Er beschreibt für den Konfigurationsmanager den Kommunikationsweg, wie die Gerätebeschreibungen zu „beschaffen“ bzw. die Zielsystemkonfigurationen zu übertragen sind. Er hat eine Interpretationsfunktion vom proprietären Gerätebeschreibungsformat in die des Zustandsmodells bzw. aus dem Zustandsmodell in das proprietäre Zielsystemformat.

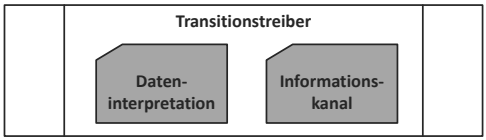

Abbildung 54: Bestandteile des Transitionstreibers

Der Transitionstreiber ist – im Gegensatz zu Gerätetreibern, wie bspw. bei USB – nicht funktions- oder geräteabhängig, sondern ist gebunden an die eingesetzten Standards und Codierungen. In Abbildung 55 ist dargestellt, wie der Transitionstreiber die Übertragungsprotokolle für das Übermitteln der Gerätedaten auf dem Konfigurationsmanager implementiert.

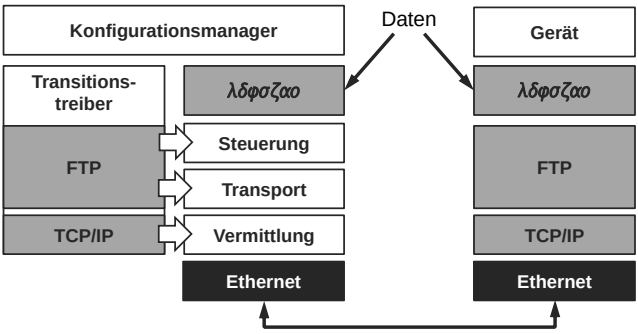

Abbildung 55: Treiber zur Herstellung der Kommunikation

Zur Herstellung der Verbindung für die Datenübertragung stellt der Transitionstreiber dem Konfigurationsmanager Funktionen zur Verfügung, die diesem Zugriff auf die Gerätebeschreibung ermöglichen. Dazu wird bspw. der für das Gerät geeignete Protokollstack initialisiert und die Datenübertragung initiiert. Die Interpretationsfunktion des Treibers dient auf der einen Seite zur Interpretation der Daten der proprietären Gerätebeschreibung, um diese Informationen abstrahiert in dem Zustandsmodell abzulegen. Auf der anderen Seite können die Informationen des Zustandsmodells mit dem Transitionstreiber in das spezifische Format des Zielsystems überführt werden. Abbildung 56 zeigt die Informationsverarbeitungskette der Übersetzerfunktion.

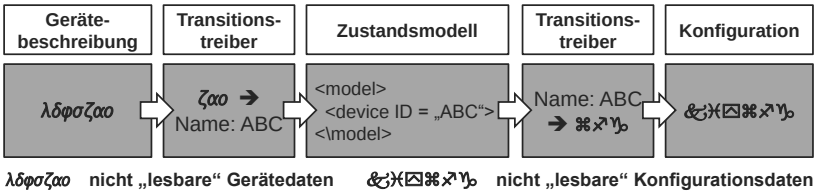

Abbildung 56: Darstellung der Dateninterpretation durch Transitionstreiber

Nicht interpretierbare Daten (Abbildung 56 links) werden durch den Treiber übersetzt und damit dem System gegenüber „verständlich“ gemacht. Im Zustandsmodell werden diese Informationen in die entsprechende Struktur integriert und damit in der Datenstruktur mit Metainformationen versehen. Dies ermöglicht bei der Konfiguration, die Informationen des Zustandsmodells in entsprechende Daten für das Zielsystem umzuwandeln (Abbildung 56 rechts).

Der Transitionstreiber erfüllt seine Aufgaben, indem er dem Konfigurationsmanager die Funktionalitäten zur Datenübertragung und -interpretation zur Verfügung stellt. In diesen Softwarefunktionen wird bspw. der erforderliche Protokollstack implementiert und die Datenübertragung angestoßen. Des Weiteren beinhalten diese Funktionen die erforderlichen Regeln zur Dateninterpretation. Diese sind ähnlich aufgebaut, wie ein Dateikonverter und überführen das Ausgangsformat in ein anderes Format. Je nachdem, ob es ein Transitionstreiber zur Informationsgewinnung oder zur Konfiguration ist, ermöglichen sie eine Konvertierung der Gerätedaten in das Zustandsmodell oder des Zustandsmodells in das Format der Zielkonfiguration.

6.5 Methodenablauf

6.5.1 Überblick über die Konfigurationssequenz

Der Beschreibung der Informationsverarbeitung folgt die Darstellung des Ablaufes der Methode während des Konfigurationsvorgangs. Dieser Ablauf besteht aus fünf wesentlichen Schritten (vgl. Abbildung 57): (1) Physikalische Verbindung; (2) Geräteerkennung; (3) Basiskommunikation; (4) Fähigkeitenerkennung; (5) Konfigurationsimplementierung. (Reinhart et al. 2010)

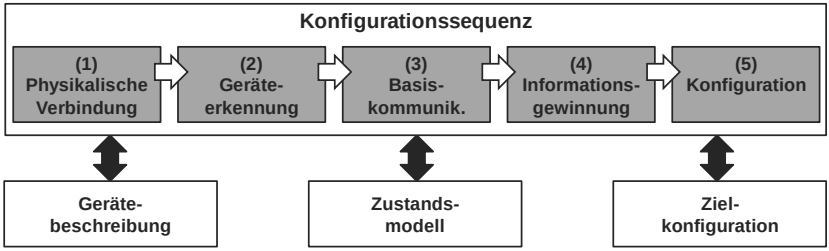

Abbildung 57: Konfigurationssequenz

Die Schritte der Konfigurationssequenz werden mithilfe der im vorangegangenen Kapitel beschriebenen Methoden zur Informationsverarbeitung durchgeführt. Im Folgenden sind die Abläufe dieser Schritte detailliert erläutert und exemplarisch anhand der Konfiguration der Robotersteuerung beschrieben. Das Vorgehen ist analog übertragbar auf die Konfiguration der verbleibenden Zielsystemarten (Programmierumgebung und Dokumentation). Sie unterscheiden sich lediglich bei der Konfigurationsübermittlung in Schritt 5.

6.5.2 Ausgangssituation des Konfigurationsablaufs

Abbildung 58 zeigt den Initialzustand bei der automatischen Konfiguration für eine Industrieroboterssteuerung und ein Peripheriegerät. Auf der rechten Seite der Abbildung ist das Peripheriegerät mit seinem Echtzeit- und Nicht-Echtzeit-Protokollstack dargestellt. Der Echtzeit-Protokollstack des Gerätes umfasst die Applikationsfunktionen (z. B. *greifer_oeffnen()*), die Prozessdaten

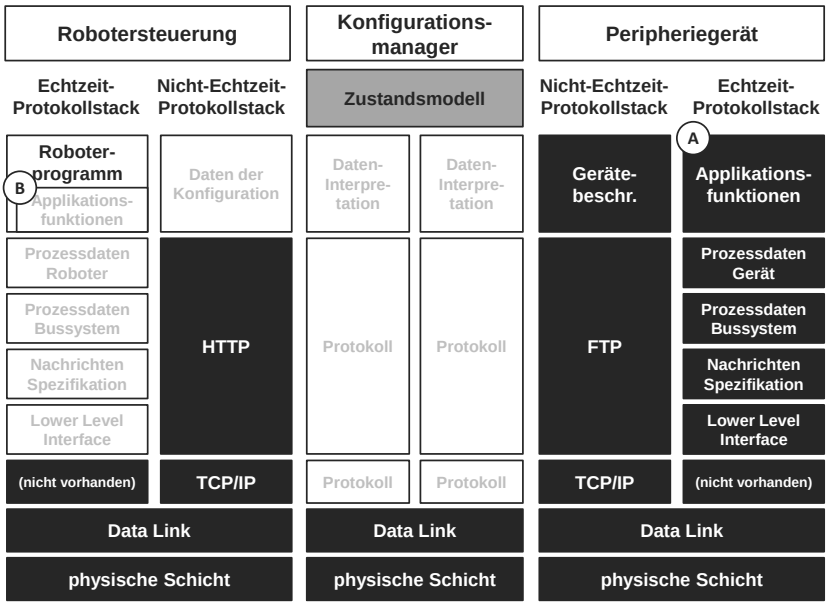

Abbildung 58: Ausgangssituation bei der automatischen Konfiguration

des Roboters, die Prozessdaten für das Bussystem, die Spezifikation der Nachricht sowie die durch das Bussystem festgelegten unteren Schichten (Data Link und die physikalische Schicht). Der nicht-echtzeitfähige Protokollstack für die Nachrichtenübertragung beinhaltet neben der Gerätebeschreibung Standard-Ethernet-Protokolle zur Datenübertragung, hier beispielhaft TCP/IP und FTP. Bei der Robotersteuerung sind die wesentlichen Schichten des Echtzeit-Protokollstacks sowie die Applikationsfunktionen nicht definiert (weiße Blöcke). Sie sind Gegenstand der Konfiguration. Lediglich die durch die Wahl des Bussystems vorgegebenen unteren Schichten sind bereits bekannt. Die nicht-echtzeitfähige Kommunikation ist durch das Betriebssystem der Geräte und der Robotersteuerung definiert. Der Konfigurationsmanager ist mit dem industriellen Netzwerk verbunden. Dadurch stehen ebenfalls die hardwarenahen Schichten des Bussystems fest. Ein „leeres“ Zustandsmodell ist vorhanden.

Ziel der Konfiguration ist es, die Applikationsfunktionen der Geräte (A) für das Roboterprogramm (B) zur Verfügung zu stellen und über den Echtzeit-Protokollstack aufrufbar zu machen.

6.5.3 Schritt 1: Physikalische Verbindung

Abbildung 59 zeigt das Vorgehen im ersten Schritt der Konfiguration. Er zeichnet sich durch vier Elemente aus: die Analyse des Netzwerks, die Vorbereitung des Netzwerks, das Entfernen von Geräten aus dem Netzwerk und das Anschließen neuer Geräte.

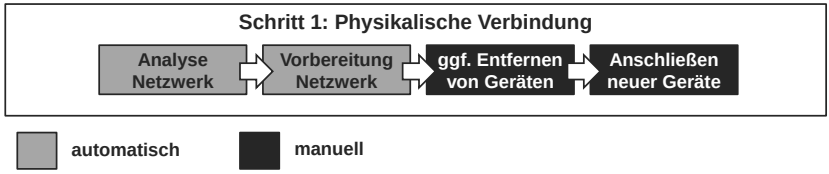

Abbildung 59: Schritt 1 – Physikalische Verbindung

Die Analyse des Netzwerks dient der Identifikation des verwendeten Netzwerkstandards, des Bussystems und der dazugehörigen netzwerknahen Übertragungsprotokolle. Mit diesen Informationen werden die nutzbaren Netzwerkmanagement-Funktionalitäten identifiziert. Bei Industrial-Ethernet-Netzwerken wird dafür der Netzwerkverkehr aufgezeichnet und analysiert: Jedes Ethernet-Paket verfügt über einen „Identifyer“, das sogenannte Typ-Feld, welches das in dem Paket verwendete Protokoll spezifiziert. Abhängig vom verwendeten Netzwerk müssen verschiedene vorbereitende Maßnahmen getroffen werden. Bei Powerlink muss bspw. die Echtzeitkommunikation gestoppt werden. Dabei ist sicherzustellen, dass die Kommunikation unterbrochen werden kann, ohne Schaden an der aufgebauten Anlage zu verursachen. Dieser Vorgang wird vom Konfigurationsmanager vorgenommen. Nicht mehr benötigte Geräte werden durch den Bediener von dem Netzwerk getrennt und aus dem Anlagenverbund entfernt. Die neuen Geräte können nun physikalisch in das Netzwerk integriert werden. Die Geräte müssen für den weiteren Konfigurationsvorgang eingeschaltet und betriebsbereit sein. Es dürfen keine Geräte mit derselben Identifikationsnummer im Netzwerk vorhanden sein.

6.5.4 Schritt 2: Geräteerkennung

Nach der physikalischen Verbindung der Geräte werden diese im Netzwerk automatisch detektiert und identifiziert (siehe Abbildung 60).

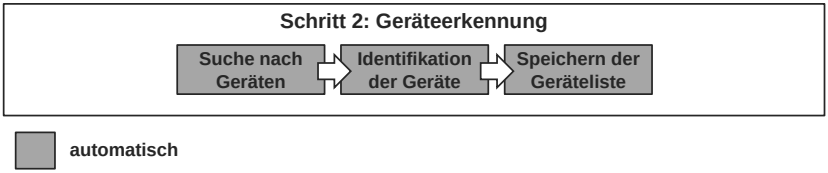

Abbildung 60: Schritt 2 – Geräteerkennung

Dies erfolgt in drei Teilschritten: die Suche nach den Teilnehmern, die Identifikation der Teilnehmer und die Erstellung einer Geräteliste. Dazu greift der Konfigurationsmanager, je nach verwendetem Kommunikationsnetz, auf verschiedene vorher identifizierte Netzwerkmanagementfunktionen zurück. Die Kommunikationsteilnehmer werden im Netzwerk gesucht. Es wird eine Kommunikationsanfrage an die Netzwerkadressen gesendet. Erfolgt eine Antwort von einer Adresse, indiziert dies die Präsenz eines Gerätes. In Industrial-Ethernet-Netzwerken mit IP-Protokoll erfolgt die Identifikation bspw. durch einen Scan der IP-Adressen. Die gefundenen Geräte werden anhand eines eindeutigen Schlüsselmerkmals (z. B. der IP-Adresse) erkannt. Es wird eine Liste der vorhandenen Geräte erstellt und gespeichert.

6.5.5 Schritt 3: Basiskommunikation

Für einen Informationsaustausch ist zunächst eine Basiskommunikation erforderlich, die es ermöglicht, auf Nicht-Echtzeit-Ebene Daten mit den Geräten und Zielsystemen auszutauschen (vgl. Abbildung 61).

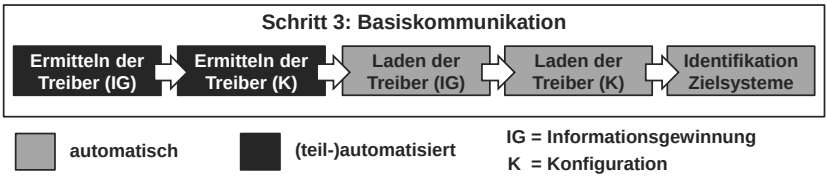

Abbildung 61: Schritt 3 – Basiskommunikation

Dieses beinhaltet das Ermitteln der Treiber sowie das Laden der Treiber für die Informationsgewinnung und Konfiguration. Nun ist dem Konfigurationsmanager

bekannt, wie die Gerätedaten zu laden und zu interpretieren sind. Des Weiteren werden die Zielsysteme definiert, für die im Folgenden die Konfiguration erstellt wird. Abbildung 62 zeigt die Protokollstacks des Konfigurationsmanagers vor und nach der Herstellung der Basiskommunikation.

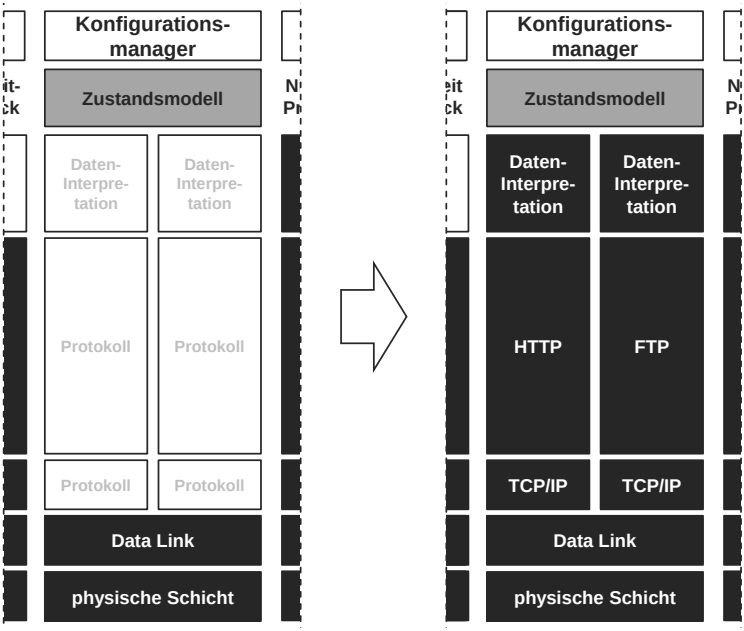

Abbildung 62: Protokollstack des Konfigurationsmanagers vor (links) und nach (rechts) dem Konfigurationsschritt 3

Zur Ermittlung der passenden Transitionstreiber für die Zielsysteme und die Geräte werden, in Anlehnung an das bestehende Verfahren der Treiber-basierten Enumeration aus der Computertechnologie (vergleiche Abschnitt 3.2.2), sequenziell drei alternative Prinzipien angewendet: Herunterladen von Gerätetreibern direkt vom Gerät über einen standardisierten Zugriff (durch die Technologievielfalt nicht immer möglich); Evaluation auf dem System verfügbarer Treiber nach der Trial-and-Error-Methode (Enumeration); manuelles Laden des geeigneten Treibers (bspw. über CD oder über Internet-Download). Derselbe Treiber kann dabei für mehrere sich von der Funktionalität unabhängige Geräte genutzt werden (vgl. Abschnitt 6.4.5 Transitionstreiber).

Die ausgewählten Treiber werden in den Konfigurationsmanager geladen. Somit wird dieser befähigt, über die spezifischen Protokolle mit den Geräten zu kommunizieren. Zudem ist durch die Transitionstreiber bekannt, wie die Geräteinformationen für die Übertragung in das Zustandsmodell zu interpretieren sind. Die ermittelten Zielsysteme – hier die Robotersteuerung – können durch die Treiberzuordnung identifiziert werden. Falls ein gewünschtes Zielsystem im Netzwerk nicht identifiziert werden kann – z. B. Dokumentation oder Offline-Programmierumgebung –, muss dieses anhand der Treiber manuell ausgewählt werden.

6.5.6 Schritt 4: Informationsgewinnung

Im Schritt Informationsgewinnung (vgl. Abbildung 63) werden die Informationen von den Peripheriegeräten in das Zustandsmodell übertragen. Hierfür sind die Informationen von den Geräten zu laden und zu speichern, die Gerätedaten in das Format des Zustandsmodells zu überführen, die geladenen Gerätebeschreibungen zu validieren und schließlich die Informationen in das Zustandsmodell einzufügen.

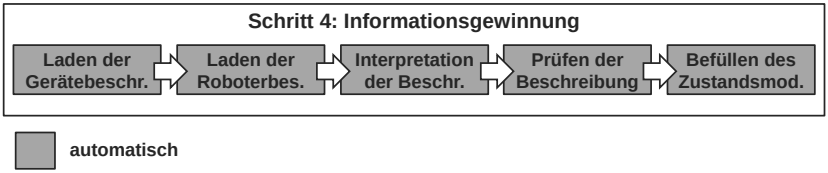

Abbildung 63: Schritt 4 – Informationsgewinnung

Durch die Definition des Protokollstacks im Transitionstreiber können die Gerätebeschreibungen auf den Konfigurationsmanager übermittelt werden. In dem hier beschriebenen Fall geschieht dies über die nicht-echtzeitfähigen Protokolle TCP/IP und FTP (vgl. Abbildung 64).

Die Informationen werden anschließend durch die Interpretationsfunktion des Transitionstreibers in das standardisierte Format des Zustandsmodells überführt und abgespeichert. Diese gespeicherte Beschreibung wird mithilfe des AutomationML-Schemas auf syntaktische und semantische Korrektheit überprüft.

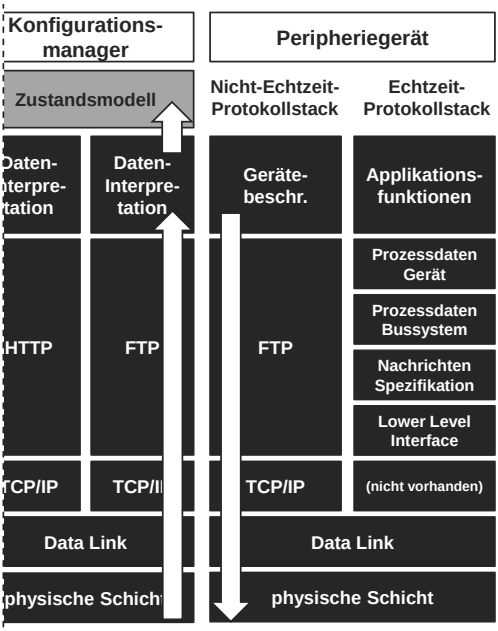

Abbildung 64: Übertragung der Gerätebeschreibung

Die Beschreibungen der Geräte und des Roboters werden in dem Zustandsmodell unter den einzelnen Informationsbereichen gespeichert. Dabei werden bestehende Freiheitsgrade und Abhängigkeiten durch den Konfigurationsmanager aufgelöst. Die Ein-/Ausgangsbelegung wird vergeben und die Verknüpfungen der Funktionen mit den Prozessvariablen durchgeführt.

6.5.7 Schritt 5: Konfigurationsimplementierung

Um die Konfiguration der Zielsysteme durchzuführen, sind die Informationen auf Vollständigkeit zu überprüfen, ggf. zu ergänzen, für die Zielsysteme in das entsprechende Format zu bringen, an das Zielsystem zu übermitteln und schließlich dort zu implementieren. Abbildung 65 zeigt eine Übersicht der erforderlichen Abläufe.
Für die gewählten Zielsysteme wird zunächst ermittelt, ob alle für die Konfiguration zwingend erforderlichen Informationen vorhanden sind. Dies ist durch

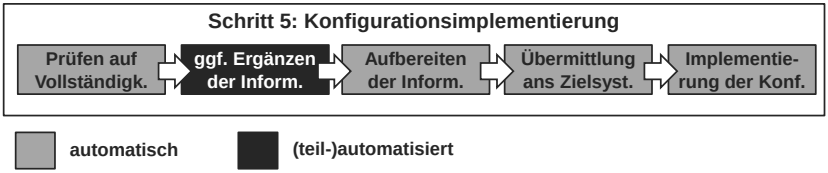

Abbildung 65: Schritt 5 – Konfigurationsimplementierung

die in den Treibern hinterlegten Informationsbedarfe möglich. Fehlen wichtige Informationen, kann eine Benutzerabfrage stattfinden, welche den Benutzer zu einer manuellen Eingabe auffordert.
Sind die Informationen vollständig, werden diese mithilfe der Dateninterpretation des Transitionstreibers in das zielsystemspezifische Format übersetzt und, soweit dies erforderlich ist, durch den ebenfalls bekannten Übertragungsweg an das Zielsystem übermittelt (vgl. Abbildung 66).

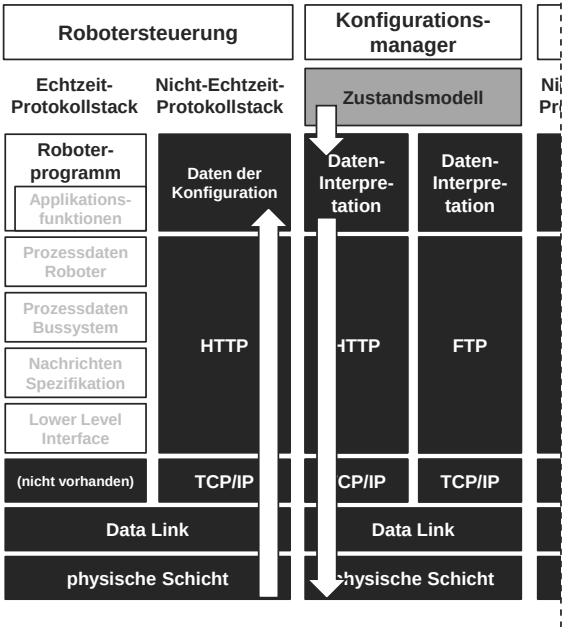

Abbildung 66: Aufbereitung und Übertragung der Konfigurationsinformationen

Diese Konfigurationsdaten müssen schließlich in dem Zielsystem implementiert werden. Dazu werden die einzelnen Schichten des Echtzeit-Protokollstacks definiert (vgl. Abbildung 67). Gegebenenfalls kann dies automatisch erfolgen. Für den Fall der Konfiguration der Robotersteuerung werden die Schichten bis hin zur Applikationsfunktion des Geräts angepasst, um die Interpretation der Prozessdaten während der Laufzeit zu ermöglichen. Ist dies nicht möglich, bspw. weil das Zielsystem Offline-Programmiersystem nicht im Netzwerk verfügbar ist, muss dieser letzte Schritt manuell durchgeführt werden. Dies kann durch einen Hinweis des Roboterherstellers im Transitionstreiber unterstützt werden.

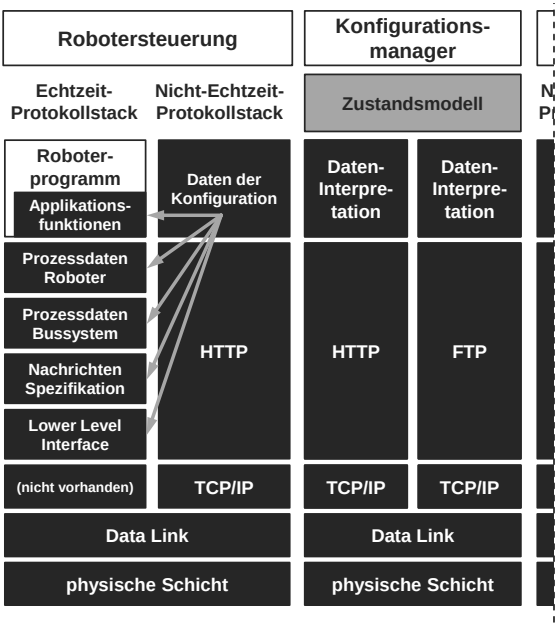

Abbildung 67: Schematische Darstellung der Implementierung der Konfigurationsinformationen im Zielsystem

Der Konfigurationsmanager sowie der nicht echtzeitfähige Protokollstack werden jetzt nicht mehr benötigt. Sie können deaktiviert oder aus dem Netzwerkverbund entfernt werden. Die Echtzeit-Kommunikation wird gestartet. Somit ist eine Kommunikation auf Applikationsebene zwischen den Geräten und dem Robotersystem möglich (vgl. Abbildung 68).

Hier differenziert sich das Vorgehen für die anderen Zielsysteme leicht. Bei der Offline-Programmierumgebung ist es hier bspw. erforderlich, Projektdateien zu speichern und diese in der Programmumgebung zu laden.

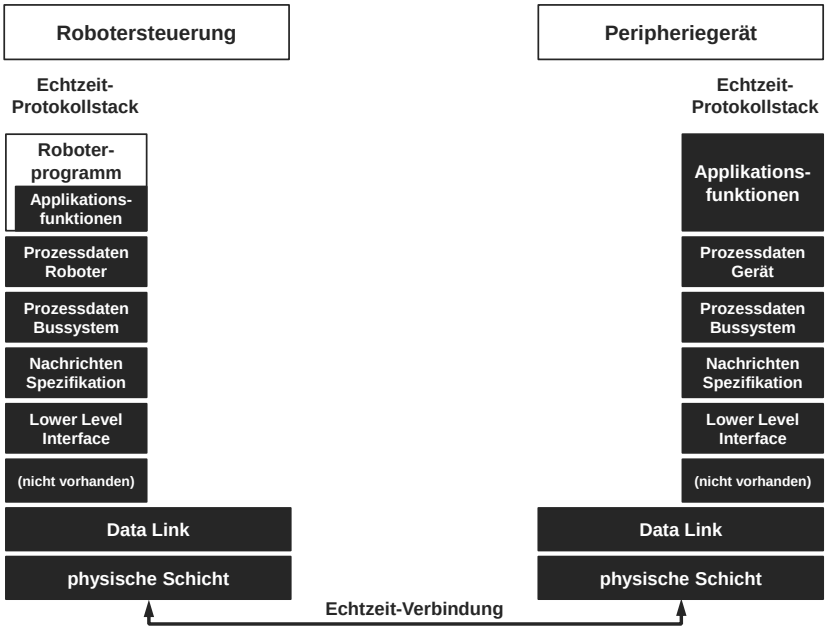

Abbildung 68: Funktionsfähige Echtzeit-Kommunikation zwischen der Steuerung des Industrieroboters und des Peripheriegerätes

6.5.8 Nutzung des konfigurierten Systems

Die bereitgestellten Gerätefunktionen können jetzt zur Erstellung des Applikationsprogramms verwendet werden. Der Programmierer wird vollständig von den darunterliegenden Kommunikationsschichten ferngehalten. Abbildung 69 zeigt die Sicht der Datenübertragung für den Applikationsprogrammierer (Ersteller des Roboterprogramms).
Somit wird die Erstellung des Applikationsprogramms, des Projektes für die Offline-Programmierung sowie für die Dokumentation deutlich vereinfacht, da die Funktionen und Eigenschaften der Geräte nicht aufwendig programmiert werden müssen. Während des Betriebs des Robotersystems sind die

Abbildung 69: Sicht des Applikationsprogrammierers

konfigurierenden Systeme nicht aktiv. Die Zuverlässigkeit des Gesamtsystems unterscheidet sich somit nicht von der Zuverlässigkeit bei einem konventionellen Konfigurationsvorgehen.

6.6 Referenzarchitektur für Plug&Produce-Softwaresysteme

6.6.1 Gestaltungsprinzipien für Softwarearchitekturen

Zur effizienten Anwendung der vorgestellten Plug&Produce-Methode in Robotersystemen inklusive der Treiber und des Zustandsmodells ist eine strukturierte Softwarearchitektur von entscheidender Bedeutung. Die nachfolgenden Absätze beschreiben das Design einer Referenzarchitektur für den Konfigurationsmanager.

In der Literatur sind verschiedene Gestaltungsprinzipien für Softwarearchitekturen zu finden. Aufgrund der strukturellen Flexibilität wurde das Gestaltungsprinzip des logischen schichtbasierten Designs (*Logical Layered Design*) für den Entwurf gewählt.

Im Folgenden ist dieses Designprinzip nach Hill (2009) beschrieben. Es sieht eine aus einzelnen Komponenten bestehende Architektur vor. Durch die sinnvolle Organisation und Abgrenzung dieser Komponenten sollen spezifische Funktionalitäten gekapselt werden. Sie sollen nach Bereichen gegliedert und über Schnittstellen miteinander verknüpft werden. Das Logical Layered Design teilt die Software in die Bereiche *Geschäftsschicht* (Business Layer), *Datenschicht* (Data Layer) und *Präsentationsschicht* (Presentation Layer).

- Die *Geschäftsschicht* implementiert die Kernfunktionalität des Systems. Sie definiert den strukturellen Programmablauf in der *Applikationslogik* (Application Logic) und kapselt die detaillierten Prozesse in der *Geschäftslogik* (Business Logic). Die Komponenten, Applikationslogik und Geschäftslogik, bieten Dienste an, welche von anderen Komponenten aufgerufen werden können.
- Die *Datenschicht* stellt den Zugang zu Datenbanken oder anderen Datenspeichern, auch bspw. über Netzwerke, her.
- Die *Präsentationsschicht* enthält nutzerorientierte Funktionalitäten, welche für die Interaktion mit dem System verantwortlich sind.

Für die jeweiligen Bereiche existieren Gestaltungsrichtlinien, welche bei der Strukturierung und der Programmerstellung berücksichtigt werden sollen. Beispiele hierfür sind das *Single Responsibility Principle* – jede Komponente soll für nur einen spezifischen Anwendungsbereich verantwortlich sein – bzw. das *Loose Coupling* – überall dort, wo es machbar ist, sollen Schnittstellenklassen definiert werden, welche die Verbindungen zwischen zwei Komponenten abstrahieren.

6.6.2 Referenzarchitektur des Konfigurationsmanagers

Für den Entwurfsprozess der Referenzarchitektur wurde die Vorgehensweise zum Design von Schichtenstruktur nach Hill (2009) angewendet. Abbildung 70 zeigt die daraus resultierende Architektur.

Geschäftsschicht

Kern der Architektur ist die Applikationslogik, welche sich in der Geschäftsschicht befindet. In dieser Komponente ist der Ablauf des fünfstufigen Konfigurationsvorgangs definiert. Sie koordiniert den Informationsaustausch mit den verschiedenen Geschäftslogiken und kommuniziert mit der Präsentationsschicht.

Aufgrund der treiberbasierten Methode wurde die Geschäftslogik in drei Bereiche aufgeteilt. Die statische Geschäftslogik beinhaltet alle gleichbleibenden Softwarefunktionen. Maßgeblich bildet sie die Logik des Zustandsmodells ab, koordiniert dessen Datenflüsse und prüft auf Vollständigkeit sowie auf syntaktische

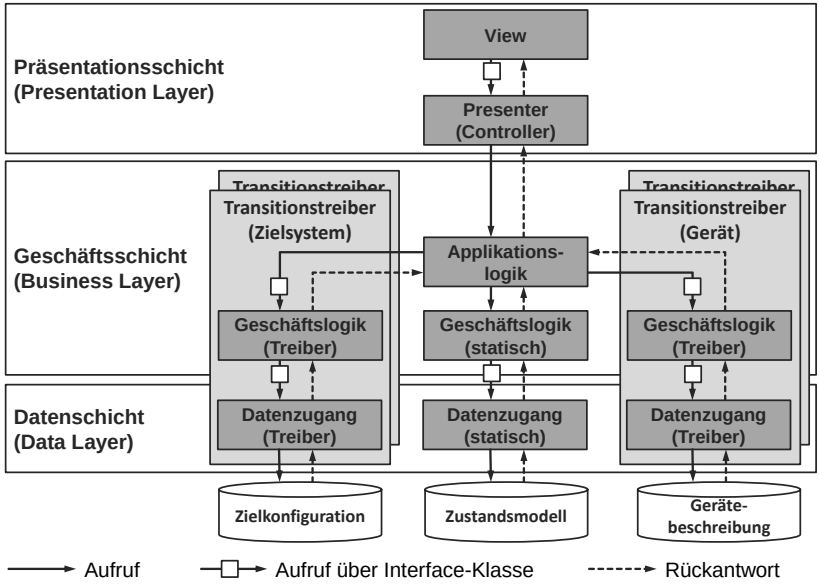

Abbildung 70: Softwarereferenzarchitektur des Konfigurationsmanagers

bzw. semantische Korrektheit. Die beiden anderen Bereiche der Geschäftslogik sind in den Treibern (zur Informationsgewinnung einerseits und zur Konfigurationsimplementierung andererseits) angesiedelt, die während des Betriebs (gegebenenfalls auch mehrfach) geladen werden. Die treiberindividuelle Geschäftslogik fasst vornehmlich die Funktionalitäten zur Dateninterpretation des Treibers (vgl. Abschnitt 6.4.5) zusammen.

Datenschicht

Ebenso wie die Geschäftslogik ist auch der Datenzugang (Data Access) untergliedert. Der statische Datenzugang ermöglicht den Zugriff auf das Zustandsmodell, seine Attribute und Daten. Der treiberindividuelle Datenzugang spezifiziert die Protokolle für den Zugriff auf die Daten der Peripheriegeräte sowie der Zielsysteme und umfasst damit die Beschreibung des Informationskanals (vgl. Abschnitt 6.4.5). Die Treiberfunktionalität ist vollständig mit der jeweiligen Geschäftslogik und dem dazugehörigen Datenzugang abgedeckt.

Präsentationsschicht

Die Präsentationsschicht wurde nach dem *Model-View-Presenter-Design-Pattern* entworfen (Boodhoo 2006). Die Daten- und die Geschäftsschicht stellen dabei das Model dar. Die View ist die grafische Darstellung (Graphical User Interface – GUI) für den Nutzer. Der Presenter beinhaltet die Logik für die Darstellung. So kann für sich verändernde Views, z. B. für eine differenzierte Darstellung für Laie und Experte, der Presenter wiederverwendet werden.

6.7 Zusammenfassung

Die Abstraktion der Datenhaltung in dem Zustandsmodell, kombiniert mit dem treiberbasierten Kommunikations- und Interpretationskonzept, führt zu einer Unabhängigkeit der Konfigurationsfähigkeit von herstellerspezifischen Standards und dem heterogenen Systemumfeld. Eine Unterstützung beliebiger Gerätefunktionalitäten wird damit gewährleistet. Die Anwend- und Übertragbarkeit auf beliebige Industrieroboter und andere Zielsysteme ist mit diesem Konzept ebenfalls sichergestellt. Die Methode kapselt die Kommunikationsebenen bis zu der funktionalen Geräteschnittstelle. Es wird kein tiefes systemspezifisches Wissen vom Bediener benötigt. Der automatische Konfigurationsvorgang findet vollständig außerhalb des Echtzeitbetriebs statt. Die für die Konfiguration erforderlichen Komponenten und Mechanismen sind während des Betriebs nicht aktiv und können aus dem Anlagenverbund entfernt werden. Dies stellt die uneingeschränkte Zuverlässigkeit und Echtzeitfähigkeit während des Betriebs sicher.

Damit ist es erstmals möglich, die Konfiguration von Robotersystemen für individuelle Funktionsumfänge bis zur Ebene der Applikationsfunktionen zu abstrahieren und zu automatisieren sowie die Erstellung von Offline-Programmierprojekten und die Dokumentation zu unterstützen.

Die vorgestellte Referenzarchitektur ist prozessorientiert und unterstützt damit die Struktur und den Ablauf der Methode. Sie bietet die Grundlage für die Umsetzung und Validierung.

7 Experimentelle Umsetzung und Validierung

7.1 Allgemeines

Dieses Kapitel beschreibt die Umsetzung und Validierung der Methode anhand eines industrienahen Anwendungsbeispiels. Zunächst wird die Versuchsumgebung mitsamt dem eingesetzten Roboter und den Peripheriegeräten beschrieben. Anschließend wird dargestellt, wie diese gemäß dem Vorgehen der funktionsorientierten Modularisierung umgestaltet und mit den erforderlichen Informationen augmentiert werden. Für die softwareseitige Unterstützung der Methode in der Versuchsumgebung wurde die Referenzarchitektur des Konfigurationsmanagers für Plug&Produce-Robotersysteme prototypisch umgesetzt und die für die Geräte erforderlichen Treiber erstellt. Der automatische Konfigurationsvorgang wurde getestet und die Methode damit validiert.

7.2 Versuchsumgebung

Für die Umsetzung und Erprobung der Methode wurde ein Versuchsstand aufgebaut (vgl. Abbildung 71).

Abbildung 71: Versuchsaufbau

Auf diesem Versuchsstand können unterschiedliche Peripheriegeräte an einen Industrieroboter angeschlossen und nach der vorgestellten Methode automatisch konfiguriert werden. Der Versuchsstand besteht aus einem Industrieroboter KR 15/2 der Firma KUKA AG mit einer Steuerung der Generation C2, einem Zweiachspositionierer DKP-400 ebenfalls der Firma KUKA AG, der Schnittstelle eines Schweißgeräts und verschiedenen einfachen Peripheriegeräten (Greifer, Abstandssensor und Lichtschranke). Als Echtzeit-Kommunikationsnetz wird das Industrial-Ethernet-Bussystem Powerlink eingesetzt. Da der Industrieroboter über keine Powerlink-Schnittstelle verfügt, wurde dieser um eine Steuerung (B&R X20) erweitert, die ausschließlich als Busumsetzer dient. Für die Verortung des Konfigurationsmanagers bestehen zwei technisch gleichwertige Alternativen (siehe Abbildung 72): Der Konfigurationsmanager ist in der Robotersteuerung integriert oder er ist auf einem separaten Rechner installiert und über einen eigenen Netzwerkzugang eingebunden. Im Weiteren ist die Variante mit einem externen Konfigurationsmanager realisiert.

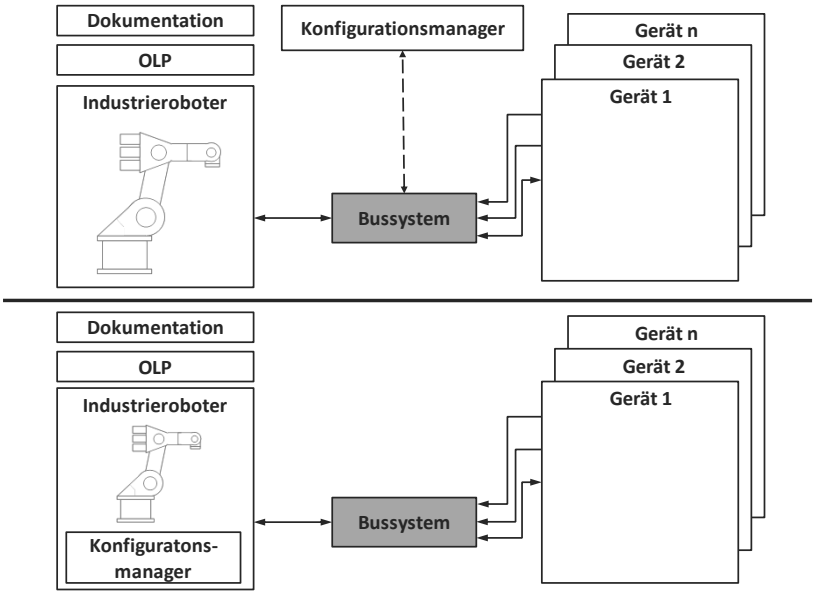

Abbildung 72: Systemarchitektur mit externem Konfigurationsmanager (oben) und in die Robotersteuerung integriertem Konfigurationsmanager (unten)

7.3 Modularisierung und Gerätebefähigung

7.3.1 Allgemeines

Exemplarisch wurden der Dreh-Kipp-Positionierer und die Schweißgerätschnittstelle für die Plug&Produce-Konfiguration befähigt. Zur Gerätemodularisierung wurde die funktionsorientierte Modularisierung nach Kapitel 5 auf die Geräte angewendet. Des Weiteren werden sie um die Fähigkeiten der Datenhaltung und der Datenübertragung (vgl. Abbildung 52) erweitert. Dabei wurde berücksichtigt, dass verschiedene Geräte unterschiedliche Fähigkeiten hinsichtlich der Signalverarbeitung, Datenhaltung und Übertragung besitzen (vgl. Abbildung 7). Die so modularisierten Geräte – Positioniertisch (intelligentes Gerät), Schweißgerät (integriertes Gerät) sowie Greifer und Lichtschranke (Basisgeräte) – wurden mit den entsprechenden Daten angereichert und zur Datenübertragung befähigt.

7.3.2 Dreh-Kipp-Positionierer DKP 400

Die folgenden Abschnitte erläutern die Umgestaltung des Dreh-Kipp-Positionierers DKP 400 der Firma KUKA, um diesen für Plug&Produce zu qualifizieren. Im Folgenden sind die durchgeführte Modularisierung und Befähigung zur Datenhaltung und -übertragung für den Dreh-Kipp-Positionierer beschrieben.

Modularisierung

Die Modularisierung des Dreh-Kipp-Positionierers wurde gemäß dem Vorgehen aus Kapitel 5 durchgeführt. Anhand des Zustandsautomaten des DKP 400 fand die Festlegung der Funktionen statt, welche das Gerät nach außen erfüllen soll. Sowohl der Zustandsautomat als auch die daraus abgeleiteten Funktionen sind in Abbildung 73 dargestellt. Die Funktionen beinhalten, soweit erforderlich, die Aufruf- und Rückgabeparameter. Eine Beispielfunktion ist *DKPinit()* für den Übergang vom inaktiven über die Referenzierung zum aktiven Zustand.

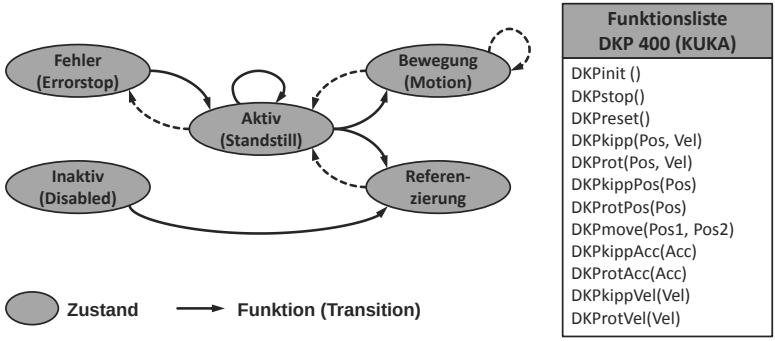

Abbildung 73: Zustandsautomat des Dreh-Kipp-Positionierers mit Zuständen und Funktionen (Transitionen)

In Abbildung 74 ist die strukturale Systemsicht des Dreh-Kipp-Positionierers nach mehreren Iterationen des Vorgehens dargestellt. Im ursprünglichen Zustand waren die Servoverstärker der Achsen in der Robotersteuerung verbaut. Der Abgleich der strukturalen und funktionalen Sicht zeigte, dass zur Funktionsdurchführung sowohl Servoverstärker als auch Steuerung innerhalb der Systemgrenze des Dreh-Kipp-Positionierers liegen müssen. Anhand des Systembilds (Abbildung 74) wurde der DKP 400 um eine B&R-Steuerung (X20) und entsprechenden Servoverstärkern erweitert. Der DKP 400 entspricht damit der funktionsorientierten Modularisierung für Plug&Produce-Robotersysteme.

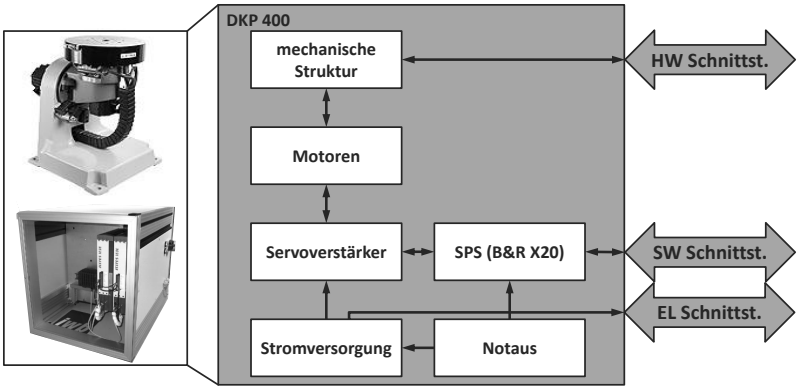

Abbildung 74: Strukturale Sicht des Dreh-Kipp-Positionierers

Datenhaltung und -übertragung

Aufgrund der eigenen Steuerung stellt der Dreh-Kipp-Positionierer ein komplexes Gerät dar. Er verfügt bereits über die Fähigkeiten der Datenverarbeitung und einer Powerlink-Netzwerkanbindung. Für die Informationshaltung wurde das vorhandene Dateisystem der X20-Steuerung genutzt. Die Informationsübertragung während der Konfiguration erfolgt über das FTP-Protokoll. Um die Informationen auf dem DKP zu hinterlegen, wurde die bestehende XDD-Datei um die erforderlichen Informationen angereichert und zudem geometrische Daten hinterlegt.

7.3.3 Schweißsteuerung (Funktionsprototyp)

Ein ähnliches Vorgehen wurde bei der Gestaltung der Schweißsteuerung angewendet. Nach der Durchführung der Modularisierung wurde ein Funktionsprototyp der Schweißsteuerung aufgebaut. Dieser stellt kein vollumfänglich funktionsfähiges Schweißgerät dar, bildet jedoch die erforderlichen steuerungstechnischen Eigenschaften zur Erprobung der Plug&Produce-Funktionalität ab. Um die Kosten für Plug&Produce-Geräte gering zu halten, soll nachgewiesen werden, dass bei deren Modularisierung sich die Fähigkeiten – Informationsübertragung und -bereitstellung – auch mit geringem Hardwareaufwand umsetzen lassen.

Der Funktionsprototyp (vgl. Abbildung 75) besteht aus einem Gehäuse mit Statusanzeigen, einem Display und einer Industrial-Ethernet-Schnittstelle. Des Weiteren besteht die Möglichkeit, zwei geometrisch unterschiedliche Schweißbrenner an das Gerät anzuschließen. Die Powerlink-Anbindung wurde mit einem konventionellen Industrial-Ethernet-Board (EDUcore-5482) der Firma SYS TEC Electronic GmbH, wie es in gängigen Peripheriegeräten zur Anwendung kommt, realisiert. Der EDUcore wurde so programmiert, dass die Funktionalitäten einer Schweißsteuerung abgebildet werden. Zur Plug&Produce-Befähigung wurde die Firmware des EDUcore so angepasst, dass eine Datenhaltung auf dem internen Speicher möglich ist. Für die Datenübertragung wurde auf dem embedded-Linux-basierten Betriebssystem ein HTTP-Server implementiert. Damit konnte gezeigt werden, dass die Datenhaltung und -übertragung für die Plug&Produce-Konfiguration mit konventionellen Industrial-Ethernet-Modulen uneingeschränkt möglich ist.

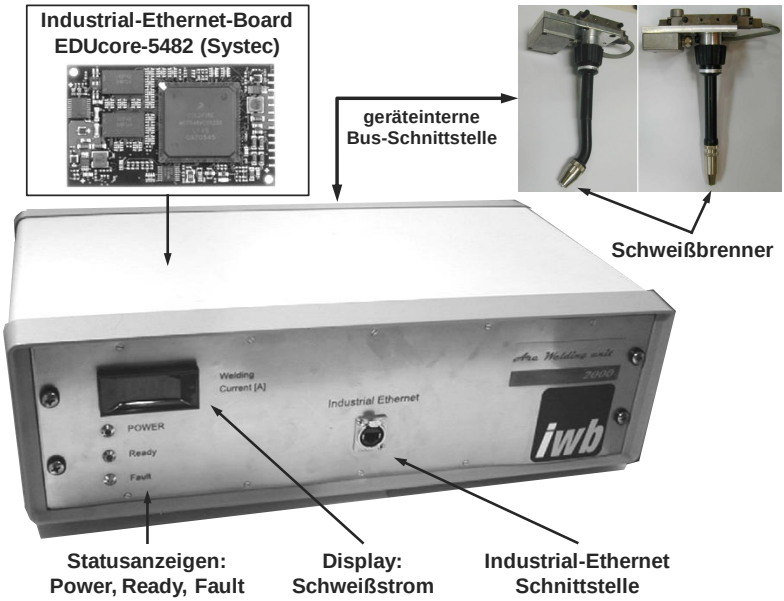

Abbildung 75: Aufbau der Schweißsteuerung

Um zudem den Nachweis zu erbringen, dass die Datenhaltung und Übertragung nicht hersteller- bzw. protokollabhängig sind, wurde die Befähigung an einem weiteren Industrial-Ethernet-Modul ausschließlich mit Powerlink-nativen Protokollen durch die Nutzung der Powerlink-Netzwerk-Managementfunktionalitäten getestet. Abbildung 76 zeigt den dafür eingesetzten Versuchsaufbau. Dieser besteht aus einem Netzteil zur Spannungsversorgung, einem konventionellen Ethernet-Switch, einem PC, auf dem die Open-Source-Software openPowerlink installiert ist, sowie dem Industrial-Ethernet-Board der Firma IXXAT Automation GmbH mit einem Altera Cyclone FPGA Chip. Auf dem Board stand neben der Speicherung des FPGA-Designs und den Programmbefehlen nicht ausreichend freier Speicher zur Verfügung, um die vollständigen Geräteinformationen zu hinterlegen. Daher wurde ein zusätzlicher 2 MB großer Flash-Speicher in das Modul integriert. Die Geräteinformationen wurden in dem sogenannten Ethernet-Powerlink-Objektverzeichnis (Object Dictionary) hinterlegt bzw. in diesem verlinkt. Mithilfe gängiger Netzwerkmanagementfunktionalitäten können Einträge und Daten des Objektverzeichnisses ausgelesen und über

das Bussystem an den PC übertragen werden. Mit einem Testprogramm, das die entsprechenden Einträge abfragt, wurde diese Variante der Übertragungsfunktionalität des Treibers getestet. In diesem Fall war die Datenhaltung und -übertragung ebenfalls möglich. Lediglich die Übertragungsgeschwindigkeit ist im Vergleich zu den Ethernet-Protokollen FTP und HTTP bei dem verwendeten Übertragungsmechanismus deutlich geringer.

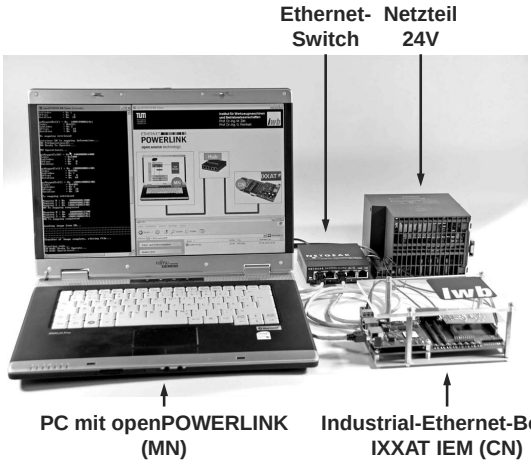

Abbildung 76: Versuchsaufbau zur protokollunabhängigen Datenübertragung

Einige Peripheriegeräte, wie Kamerasysteme, können bspw. durch Umprogrammierung ihre Eigenschaften oder den Funktionsumfang ändern. Der zielgerichtete Umgang mit solch veränderlichen Geräten soll ebenfalls anhand des Funktionsprototyps gezeigt werden. Das veränderliche Merkmal ist in diesem Fall die Geometrie der unterschiedlichen Schweißbrenner. Zu diesem Zweck wurde die Datenhaltung und Datenbereitstellung zumindest teilweise getrennt. Der Teil der Daten, die von der unterschiedlichen Geometrie beeinflusst ist, wurde direkt auf dem Schweißbrenner gespeichert. Zu diesem Zweck wurde er mit einem einfachen Speichermedium und einer eigenständigen internen Busverbindung (hier USB) ausgestattet. In der Gerätebeschreibung der Schweißsteuerung erfolgte schließlich eine Verlinkung und damit eine Aktualisierung der Geräteinformationen beim Austausch des Brenners.

7.3.4 Einfache Peripheriegeräte

Bei diesen Peripheriegeräten, wie Lichtschranken, kapazitiven Abstandssensoren, pneumatischen Zylindern etc. übersteigen selbst die Kosten für eine reine Busanbindung die Rentabilitätsgrenze. Solche einfachen Geräte werden heute mit sogenannten E/A-Modulen durch Verdrahtung an das Bussystem angebunden. Die Methode lässt sich ebenfalls auf diesen Anwendungsfall übertragen. Ebenso wie bei dem Schweißgerät wurden bei den einfachen Peripheriegeräten die Datenhaltung und -übertragung separiert. Individuelle Eigenschaften der einfachen Geräte werden auf diesen gespeichert. Zu diesem Zweck wurden Steckverbinder mit einem SD-Speicher ausgestattet (vgl. Abbildung 77) und die Informationen dort hinterlegt.

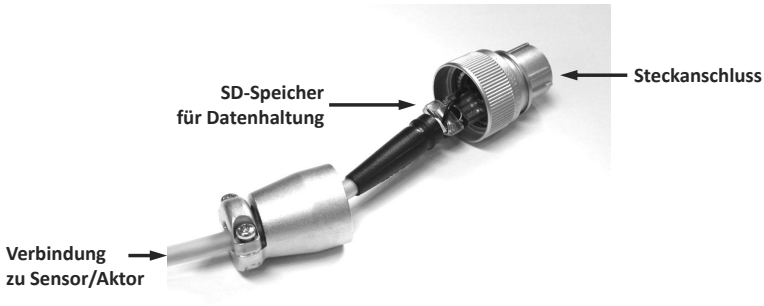

Abbildung 77: Stecker mit Datenhaltung für einfache Peripheriegeräte

Die Busanbindung und die Datenübertragung erfolgen über einen speziell entwickelten Geräte-Hub (vgl. Abbildung 78). Dieser Hub (zu deutsch Knotenpunkt) fasst die Informationen der gerade verbundenen Stecker zu einer Gerätebeschreibung zusammen. Er besteht aus einem EDUcore-5482 Board mit entsprechender Hardware zur Spannungsversorgung. Die mit Informationen angereicherten Geräte, wie z. B. eine Lichtschranke oder ein Greifer, können über einen Stecker an den Hub angeschlossen werden. Die Daten des Geräts können nun übertragen und von dem Bus-Modul im Hub zu einer Gerätebeschreibung zusammengefasst werden.

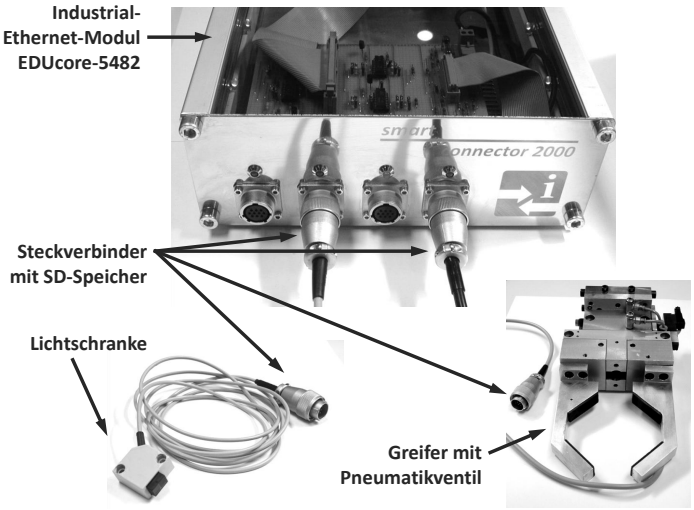

Abbildung 78: Plug&Produce-Hub für einfache Peripheriegeräte

Die Systemgrenze für funktionsorientierte Module ist dabei um den Hub inklusive der einfachen Geräte zu ziehen (vgl. Abbildung 79). Die gestrichelt eingezeichneten Verbindungspfeile stellen die Schnittstelle zu den an den Hub angeschlossenen Geräten dar. Nach dem jeweiligen Anschließen wird die Gerätebeschreibung aktualisiert.

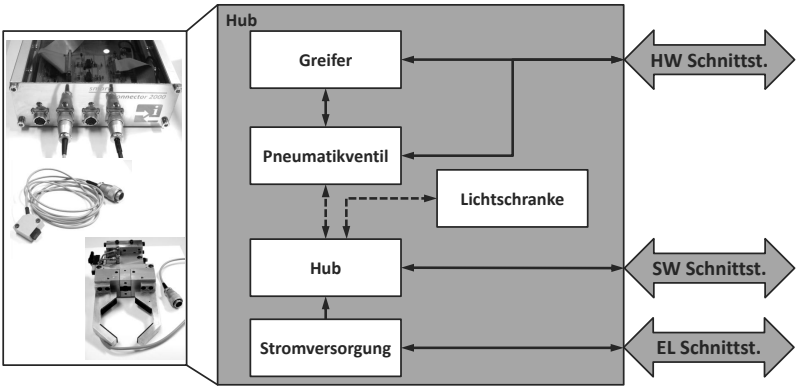

Abbildung 79: Systemgrenze des Plug&Produce-Hubs

7.4 Software-Umsetzung des Konfigurationsmanagers

Auf Basis der in Abschnitt 6.6.2 vorgestellten Referenzarchitektur wurde der Konfigurationsmanager umgesetzt.

Die jeweiligen Treiber sind von einer Treiber-Prototypenklasse abgeleitet, besitzen bereits das Interface zu der Applikationslogik sowie die wichtigen Funktionsrümpfe, um bspw. Informationen im Zustandsmodell zu hinterlegen. Die Treiber sind als Dynamic Link Libraries (DLL) konzipiert, die während der Ausführung des Programms dynamisch geladen werden.

Für die Bedienung der Software wurden zwei verschiedene Benutzeroberflächen implementiert: ein Experteninterface, bei dem sich der Konfigurationsvorgang Schritt für Schritt nachvollziehen lässt und ein benutzerfreundliches Interface, welches den Konfigurationsvorgang weitestgehend vom Anwender fernhält und lediglich bei den erforderlichen manuellen Eingriffen mit dem Nutzer interagiert. Abbildung 80 zeigt das implementierte einfache Benutzerinterface vor dem Starten des Konfigurationsvorgangs.

Abbildung 80: Einfaches Benutzerinterface des Konfigurationsmanagers

Ein UML-Sequenzdiagramm, welches den Ablauf der Software (inklusive der fünf Schritte der Methode) sowie die Interaktion der einzelnen Softwarekomponenten untereinander beschreibt, findet sich in Anhang C.

7.5 Experimentelle Durchführung der Konfiguration

Zur Evaluierung der Methode wurde anhand eines Konfigurationsszenarios der Industrieroboter für eine neue Aufgabe umgerüstet. Der Roboter soll dazu befähigt werden, eine Schweißaufgabe durchzuführen. Über ein Ethernet-Powerlink-basiertes Echtzeit-Kommunikationssystem können die Schweißsteuerung und der Dreh-Kipp-Positionierer an den Industrieroboter angeschlossen werden. Der Konfigurationsmanager, der auf einem separaten Computer läuft, ist ebenfalls in das Netzwerk integriert. Die Durchführung der Konfiguration gestaltet sich einfach. Auf der Nutzeroberfläche muss der Bediener den Konfigurationsvorgang starten. Er wird aufgefordert die neuen Geräte anzuschließen und die nicht mehr benötigten aus dem Anlagenverbund zu entfernen. Der Nutzer wählt dann (aus einer Vorauswahl) das Zielsystem aus. Wenn die Geräteinformationen vollständig sind, werden die verbleibenden Schritte der Plug&Produce-Konfiguration automatisch durchgeführt. Nach einer Rückmeldung des Konfigurationsmanagers ist das System betriebsbereit. Nun kann mit der Erstellung des Applikationsprogramms begonnen werden.

Probandenstudie

Es wurde eine Probandenstudie durchgeführt, um die Konfigurationszeiten nach der Plug&Produce-Methode und nach der herkömmlichen Methode miteinander zu vergleichen.

Die Konfiguration der Robotersteuerung mit Plug&Produce-System wurde von insgesamt acht Testpersonen (vier mit und vier ohne Konfigurationserfahrung durchgeführt. Die Testpersonen mit Erfahrung benötigten dabei im Schnitt 7 Minuten und 56 Sekunden. Bei den Probanden ohne Erfahrung waren im Durchschnitt 8 Minuten und 2 Sekunden erforderlich. Der geringe Zeitunterschied zwischen erfahrenen und unerfahrenen Testpersonen belegt eine Entkopplung der benötigten Konfigurationszeit von dem Expertenwissen.

Zur Abschätzung der Dauer des konventionellen Konfigurationsvorgangs wurden die erfahrenen Probanden nach ihrer Einschätzung gefragt, wie lange sie für den konventionellen Konfigurationsvorgang benötigen. Im Mittel gaben die Testpersonen eine Dauer von 28,7 Stunden an. Zwei Testpersonen, eine mit und eine ohne Erfahrung, führten den Konfigurationsvorgang zudem nach dem konventionellen Verfahren durch. Ohne Einsatz des Plug&Produce-Systems benötigte die Testperson ohne Erfahrung ca. 70 Stunden für die Konfiguration des Dreh-Kipp-Positionierers inklusive der Erstellung der Funktionalitäten. Der die erfahrene Testperson benötigte für den gleichen Vorgang ca. 18 Stunden. Die Ergebnisse zeigen eine deutliche Reduzierung des zeitlichen Konfigurationsaufwands durch die Methode zur automatischen Konfiguration von Robotersystemen. Eine Übersicht der Ergebnisse der Probandenstudie findet sich in Anhang D.

7.6 Zusammenfassung

In diesem Kapitel wurde die Umsetzung der automatischen Konfiguration beschrieben. Es wurden folgende Peripheriegeräte nach dem vorgestellten Vorgehen modularisiert und mit zusätzlichen Daten sowie der Fähigkeit der Datenübertragung ausgestattet: Dreh-Kipp-Positionierer DKP 400, Funktionsprototyp eines Schweißgeräts und ein Hub mit entsprechenden einfachen Geräten. Ergebnis ist, dass eine Modularisierung in den vorliegenden Fällen möglich war und umgesetzt werden konnte. Es ist, mit Ausnahme der einfachen Peripheriegeräte, keine zusätzliche Steuerungshardware erforderlich, um bei den Geräten Datenhaltung und -übertragung zu implementieren. Konventionelle Industrial-Ethernet-Module konnten befähigt werden, diese Funktionalitäten zu erfüllen. Die Geräteinformationen für die vorliegenden Geräte wurden teilweise in unterschiedlichen Formaten hinterlegt. Der Umgang mit dieser Formatheterogenität war in den getesteten Fällen problemlos möglich.

Der Konfigurationsmanager wurde nach dem Vorbild der Referenzarchitektur implementiert. Dabei wurden die einzelnen Schritte der Methode softwareseitig umgesetzt. Für die unterschiedlichen Formate der Informationsbereitstellung und Konfiguration wurden Transitionstreiber geschrieben, die Software getestet und an dem industrienahen Beispiel validiert. Damit konnte die Funktionsweise der Treiber-basierten Interpretation und Übertragung in das Zustandsmodell nachgewiesen werden.

Schließlich wurde ein Industrieroboter mithilfe der neuen Methode für ein Anwendungsbeispiel konfiguriert. Die automatische Einbindung der Geräte (DKP und Schweißsteuerung) war ohne Einschränkungen möglich. Versuche mit Probanden haben eine deutliche Reduzierung des Konfigurationsaufwands gezeigt.

Die Unabhängigkeit von Beschreibungsstandards sowie die Reduzierung des erforderlichen Expertenwissens konnten nachgewiesen werden. Die experimentelle Umsetzung bestätigte die Funktionsweise der Methode zur automatischen Konfiguration.

8 Technische und wirtschaftliche Bewertung

8.1 Allgemeines

Die in den vorangegangenen Kapiteln aufgezeigte Systemgestaltung, Umsetzung und Validierung bilden die Basis für die technische, nutzerorientierte und wirtschaftliche Bewertung der Methode zur automatischen Konfiguration von Robotersystemen. Die Erkenntnisse aus dem prototypischen Aufbau werden mit den technischen Anforderungen (vgl. Kapitel 4) verglichen. Die erreichten Verbesserungen hinsichtlich Bedienbarkeit, Nutzbarkeit und des erforderlichen Expertenwissens werden aufgezeigt. Es folgt eine Betrachtung wirtschaftlicher Gesichtspunkte, indem die Aufwände dem Nutzen gegenübergestellt und die Einsatzpotenziale für Plug&Produce-Robotersysteme diskutiert werden.

8.2 Technische Bewertung

Die technische Bewertung erfolgt anhand der Anforderungen an Plug&Produce-Robotersysteme, indem die neue Methode dem konventionellen Konfigurationsprozess gegenübergestellt wird.

Die *systemtechnische Gesamtzuverlässigkeit* wird nicht eingeschränkt. Die Konfigurationsmechanismen sind von der Kommunikation während des Betriebs strikt getrennt. Durch die Passivität bzw. Abwesenheit aller Konfigurationssysteme bleibt die Zuverlässigkeit des Gesamtsystems im Produktivbetrieb erhalten.

Der Übertragungsweg der Prozessdaten wird durch diese Konfigurationsmethode nicht beeinflusst. Die *Echtzeitfähigkeit* ist im Rahmen der technischen Möglichkeiten des Bussystems uneingeschränkt gegeben.

Der *Funktionsumfang* der Peripheriegeräte ist nicht reduziert. Durch die Abstraktion der Kommunikation bis zur funktionalen Ebene ist ein beliebiger Funktionsumfang darstellbar. Dadurch wird auch die *Flexibilität* von Industrierobotern hinsichtlich ihrer Prozess- und Anwendungsbreite nicht durch die automatische Konfiguration eingeschränkt.

Durch das Treiber-basierte Konzept kann eine *Kompatibilität* zu verschiedenen Standards hergestellt werden. Dies gilt auch für die Unterstützung verschiedenster Industrieroboter und Industrial-Ethernet-basierter Kommunikationssysteme.

Die Konfiguration umfasst keine sicherheitsrelevanten Komponenten. Notausschalter, Sicherheitslichtschranken etc. müssen in konventioneller Weise integriert werden. Die *Sicherheit* wird von der Konfiguration demnach nicht beeinflusst.

Bei den Geräteherstellern ist ein Initialaufwand zu betreiben, die Geräte funktionsorientiert zu modularisieren, die Daten zu hinterlegen und ggf. für einen spezifischen Standard einen Treiber zu erstellen.

Der *Konfigurationsaufwand* konnte stark reduziert werden. Waren bisher teilweise mehrere Tage erforderlich, um die Konfiguration von Robotersystemen durchzuführen, ist es mit der neuen Methode möglich, die Einbindung der Geräte in wenigen Minuten vorzunehmen.

8.3 Nutzerorientierte Bewertung

Im Vergleich zum derzeit praktizierten Konfigurationsprozess werden mit dieser neuen Methode folgende Vorteile für den Anwender erzielt:

Aufgrund der hohen Abstraktion der Kommunikation wird im Vergleich zum bestehenden Vorgehen kaum systemtechnisches Wissen vom Benutzer gefordert.

Eine intuitive, situativ gestaltete Benutzerführung während des Konfigurationsvorgangs unterstützt die Einfachheit der Bedienung. Der Nutzer kann sich auf den Produktionsprozess fokussieren.

Durch die Reduktion des Konfigurationsaufwands ist es möglich, Geräte einfacher auszutauschen. Der Einsatz von Spezialisten für die Rekonfiguration von Anlagen kann so reduziert werden.

8.4 Wirtschaftliche Bewertung

Die wirtschaftlichen Betrachtung erfolgt, ähnlich wie bei Werkzeugmaschinen, auf Basis von Maschinenstundensätzen. Für die Ausstattung eines Robotersystems mit Plug&Produce-Fähigkeiten entstehen Kosten, welche im Vergleich den Maschinenstundensatz des Robotersystems erhöhen. Bei dem konventionellen Vorgehen entstehen bei jeder Planung, Systemintegration und Programmierung hohe Kosten. Zur Sicherstellung der Wirtschaftlichkeit muss die Kosteneinsparung durch den reduzierten Konfigurationsaufwand höher sein, als die Mehrbelastung durch die höheren Maschinenkosten. Im Detail lassen sich nach Reinhart & Krug (2009) die Kosten für ein rekonfigurierbares Robotersystem in zwei Kostenarten unterteilen: Stückkosten (variabler Kostenanteil, wie bspw. der Maschinenstundensatz pro Produkt) und Rüstkosten für die Rekonfiguration. Betrachtet man nun die Stückkosten für z. B. die Roboternutzung, Stellfläche und Bedienpersonal (in $S_{P\&P}$ und $S_{konventionell}$ zusammengefasst) und Systemintegration und Programmierung als Rüstkosten ($K_{rüst\text{-}P\&P}$ und $K_{rüst\text{-}konventionell}$), ergibt sich ein qualitativer, stückzahlabhängiger Kostenverlauf ($S+K/n$) für ein konventionelles Robotersystem und ein Plug&Produce-fähiges Robotersystem (vgl. Abbildung 81).

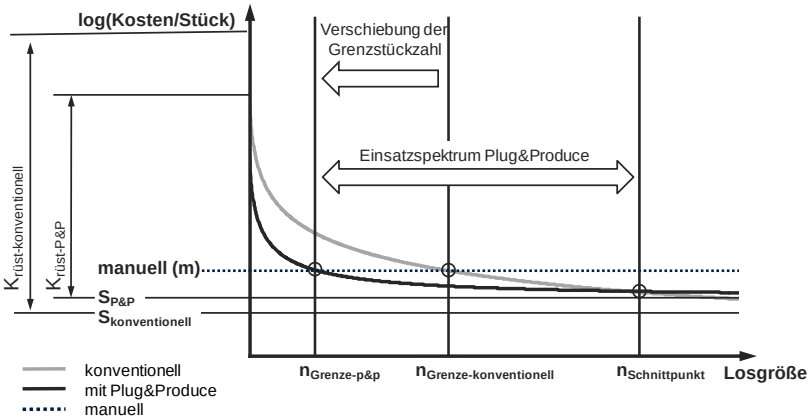

Abbildung 81: Berechnungsgrundlage für Wirtschaftlichkeit in Anlehnung an Reinhart & Krug *(2009)*

Zudem zeigt die Grafik die Kosten für die manuelle Durchführung des Arbeitsschrittes als ausschließlich variable Kosten (m).

Der Schnittpunkt der Kurven mit den manuellen Herstellkosten liefert je eine Grenzstückzahl, ab der sich der Einsatz eines Industrieroboters im Vergleich zur manuellen Tätigkeit lohnt.

$$n_{\text{Grenze}} = \frac{K_{\text{rüst}}}{(m - S)}$$

mit n_{Grenz} als Grenzstückzahl, $K_{rüst}$ als Rüstkosten, m als Kosten für die manuelle Herstellung und S als Stückkosten (Reinhart & Krug 2009)

Wie aus Abbildung 81 ersichtlich ist, reduziert sich die Grenzstückzahl für einen wirtschaftlichen Robotereinsatz durch die Verwendung der Plug&Produce-Methode, indem die Rüstkosten durch die Reduzierung des Konfigurationsaufwands sinken. Der einfach zu rekonfigurierende Industrieroboter ist bis zu einer gewissen Stückzahl $n_{\text{Schnittpunkt}}$ gegenüber dem konventionellen wirtschaftlicher.

$$n_{\text{Schnittpunkt}} = \frac{K_{\text{rüst-konventionell}} - K_{\text{rüst-p\&p}}}{S_{\text{p\&p}} - S_{\text{konventionell}}}$$

mit $K_{rüst\text{-}konventionell}$ und $K_{rüst\text{-}p\&p}$ als die jeweiligen Rüstkosten sowie $S_{konventionell}$ und $S_{p\&p}$ als die jeweiligen Stückkosten

Durch die geringeren Kosten im Vergleich zu konventionellen Robotersystemen und der manuellen Durchführung sind Plug&Produce-Robotersysteme im Bereich zwischen $n_{\text{Grenze-p\&p}}$ und $n_{\text{Schnittpunkt}}$ im Vergleich zu konventionellen Lösungen wirtschaftlicher einsetzbar.

8.5 Zusammenfassung

Durch Plug&Produce-Robotersysteme nach der hier beschriebenen Methode können das erforderliche Expertenwissen zur Konfiguration sowie der dafür benötigte Aufwand, bei gleichzeitiger Erhaltung der Alleinstellungsmerkmale und erforderlichen Eigenschaften von Robotersystemen, stark reduziert werden. Abbildung 82 zeigt einen qualitativen Vergleich der bewerteten Kriterien von der Konfiguration mit und ohne Plug&Produce-Methode.

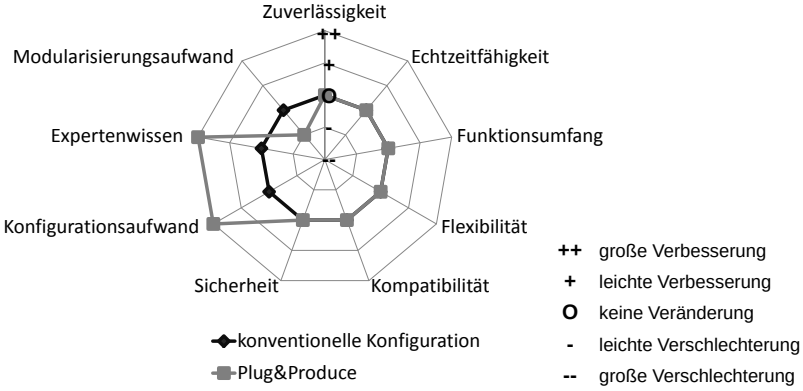

Abbildung 82: Zusammenfassung der technischen und nutzerorientierten Bewertung

Die automatische Konfiguration von Robotersystemen ist in der Lage, die Grenzstückzahl für einen wirtschaftlichen Einsatz von Industrierobotern zu senken. Gleichzeitig weist sie bei kleineren Stückzahlen einen wirtschaftlichen Vorteil gegenüber konventionell konfigurierten Systemen auf.

9 Zusammenfassung und Schlussbetrachtung

Das von kleiner werdenden Losgrößen geprägte produktionstechnische Umfeld erfordert neue und innovative Ansätze in der Robotik, die dort einen wirtschaftlichen Einsatz ermöglichen. Die kostenträchtige Systemintegration der Roboter und der dazugehörigen Peripheriegeräte stellt dabei einen wesentlichen Hebel dar, die Wandlungsfähigkeit dieser Systeme und damit auch deren Einsatz in einer flexiblen Fertigung bei kleinen und mittleren Stückzahlen zu erhöhen.

Bisher waren keine umfassenden Methoden verfügbar, die eine einfache Konfiguration unter industriellen Rahmenbedingungen ermöglichen. Bestehende Ansätze beschränken sich auf spezifische Funktionsumfänge der Peripheriegeräte, setzen auf einengende standardisierte Protokolle oder fokussieren nicht die Spezifika von Industrierobotern und deren Umfeld. Die derzeitigen Entwicklungen in der Automatisierungstechnik – Trend zur Modularität, zunehmend verteilte Steuerungsarchitekturen und Ethernet-basierte Bussysteme – schaffen neue Möglichkeiten den Aufwand für die Konfiguration zu reduzieren.

Diese Arbeit beschreibt eine neuartige Methode zur automatischen Konfiguration (Plug&Produce) von Industrierobotern. Ausgehend von einer Analyse des Informationsflusses während einer konventionellen Konfiguration, werden die Anforderungen an automatisch konfigurierbare Robotersysteme und der entsprechende Informationsbedarf erörtert. Ein Vorgehen zur funktionsorientierten Modularisierung der Peripheriegeräte wird vorgestellt. Damit ist eine systematische Anleitung gegeben, um die systemtechnischen Anforderungen an Plug&Produce-fähige Geräte zu erfüllen.

Auf Basis allgemeiner kommunikations- und systemtechnischer Analysen wurde eine Methode zur automatischen Konfiguration abgeleitet. Diese erlaubt mit einem Zustandsmodells und den entsprechenden Treibern eine automatische Konfiguration im heterogenen Umfeld der Industrieroboter, Peripheriegeräte, Bussysteme und deren jeweiligen Standards. Innerhalb eines fünfstufigen Vorgehens wurde der detaillierte Ablauf der Methode vom Anschließen der Geräte bis hin zur vollständigen Konfiguration der Robotersteuerung, der Programmierumgebung und der Dokumentation beschrieben. Für den informationstechnischen

Aufbau wurde eine Software-Referenzarchitektur entwickelt, die eine individuelle Umsetzung im jeweiligen Einsatzgebiet erlaubt. Zur Validierung wurde der Ablauf der Modularisierung an verschieden komplexen Peripheriegeräten getestet. Die Methode zur automatischen Konfiguration wurde prototypisch implementiert und anhand eines industrienahen Produktionsszenarios abgesichert. Dabei konnte sowohl eine deutliche Reduzierung der Konfigurationszeit als auch des erforderlichen Expertenwissens nachgewiesen werden. Eine technisch wirtschaftliche Bewertung steckt den sinnvollen und vor allem wirtschaftlichen Einsatzrahmen von Plug&Produce-Robotersystemen nach der vorgestellten Methode ab.

Die Ergebnisse dieser Arbeit zeigen die Möglichkeiten und das Nutzenpotenzial der automatischen Konfiguration von Robotersystemen ins Besondere für kleine und mittlere Losgrößen. Die realitätsnahe Umsetzung ermöglicht eine einfache Übertragung in die industrielle Anwendung.

Für einen nachhaltigen Erfolg dieser Technologie ist ein Paradigmenwechsel bei dem Einsatz von Industrierobotern erforderlich. Die Entwicklung einer Roboterlösung darf nicht wie bisher in langwierigen Planungsprozessen erfolgen, in denen die Anwendung bis ins letzte Detail vorgedacht und der Roboter nach seinem Einsatz nicht mehr wiederverwendet wird. Vielmehr muss sich eine Nutzung des Roboters ähnlich einer Werkzeugmaschine einstellen: Der Werker vor Ort richtet den Roboter für eine bestimmte Aufgabe ein und der Roboter wird danach direkt für den nächsten Einsatz weiterverwendet. Auch die Kalkulation zur Anschaffung von Industrierobotern muss sich dahin gehend grundlegend ändern. Der Roboter wird sich nicht mehr in einem einzigen Einsatzszenario amortisieren. Es müssen Maschinenstundensätze als Grundlage für die Kostenkalkulation herangezogen werden.

Literaturverzeichnis

Abele & Reinhart 2011

Abele, E.; Reinhart, G.: Zukunft der Produktion – Herausforderungen, Forschungsfelder, Chancen. München: Carl Hanser 2011.

Armbruster et al. 2006

Armbruster, H.; Kirner, E.; Kinkel, S.: Neue Nutzungspotentiale für Industrieroboter – Ergebnisse einer Betriebsbefragung. wt Werkstattstechnik online 96 (2006) 9, S. 631–636.

Axelson 2009

Axelson, J.: USB complete – The developer's guide. 4. Auflage. Madison, Wisconsin (USA): Lakeview Research 2009.

Barnes & Levy-Finch 2008

Barnes, M.; Levy-Finch, E. (Hrsg.): COLLADA™ - Digital Asset Schema Release 1.5.0. Khronos Group and Sony Computer Entertainment 2008.

Bartscher 2011

Bartscher, S.: Mensch-Roboter-Kooperation in der Produktion – Chancen und Risiken. Robot World 6 (2011) 1/2, S. 8–9.

Baumann et al. 2009

Baumann, M.; Feike, M.; Neukäufer, M.; Okon, M.; Popova, N.; M., S.; Schneickert, S.; Wessner, M.: Abschlussbericht Verbundprojekt ProduFlexil, Karlsruher Institut für Technologie (KIT), 2009.

Bengel 2008

Bengel, M.: Modelling Objects for Skill-Based Reconfigurable Machines. In: Pham, D. T.; Eldukhri, E. E.; Soroka, A. J. (Hrsg.): Innovative production machines and systems – 3rd I*PROMS Virtual International Conference 2007. 02.-13. Juli 2007. Dunbeath (Schottland): Whittles Publishing 2008. S. 238–243.

Bengel & Pflüger 2008
Bengel, M.; Pflüger, M.: Cell-Controller für rekonfigurierbare Produktionsanlagen – Durchgängige Prozesskette von Benutzereingabe bis Maschinenebene. In: 5. Fachtagung Robotik 2008: Leistungsstand – Anwendungen – Visionen – Trends. München, 11.-12. Juni 2008. Düsseldorf: VDI-Verlag 2008, (VDI-Berichte 2012). S. 273–276.

Bittner & Spence 2003
Bittner, K.; Spence, I.: Use Case Modelling. Boston, Massachusetts (USA): Pearson Education 2003.

Boodhoo 2006
Boodhoo, J.-P.: Design Patterns – Model View Presenter. MSDN Magazin (2006) 8, Elektronische Ausgabe.

Boterenbrood 2000
Boterenbrood, H.: CANopen – High-Level Protocol for CAN-Bus – Version 3.0. Technische Spezifikation. Amsterdam (Niederlande): NIKHEF 2000.

Bär et al. 2008
Bär, T.; Mandel, S.; Sauer, O.; Ebel, M.: Durchgängiges Datenmanagement durch plug-and-work zur virtuellen Linieninbetriebnahme. In: 2. Karlsruher Leittechnisches Kolloquium 2008. Karlsruhe, 28.-29. Mai 2008. Stuttgart: Fraunhofer IRB Verlag 2008. S. 105–121.

Brecher et al. 2008
Brecher, C.; Herfs, W.; Jensen, S.; Kolster, D.; Preßow, M.: "Plug&Play" – eine Vision rückt näher: Sind selbstkonfigurierende Visualisierungen von komponentenbasierten Automatisierungsanlagen schon bald Realität? In: A&D-Kompendium 2008/2009, München: publish-industries 2008. S. 26–29.

CCC 2009
CCC: From Internet to Robotics: A Roadmap for US Robotics. Washington D.C. (USA): Computing Community Consortium and Computing Research Association 2009.

DIN 11593
DIN 11593: Automatische Wechselsysteme für Endeffektoren. Berlin: Beuth 1996.

DIN 16030

DIN 16030: Fluidtechnik – Pneumatik-Leitungsanschlüsse. Berlin: Beuth 2005.

DIN 60038

DIN 60038: CENELEC-Normspannungen. Berlin: Beuth 2011.

DIN 60320

DIN 60320: Gerätesteckverbindungen für den Hausgebrauch und ähnliche allgemeine Zwecke. Berlin: Beuth 2008.

DIN 61158

DIN 61158: Industrielle Kommunikationsnetze – Feldbusse. Berlin: Beuth 2011.

DIN 61355

DIN 61355: Klassifikation und Kennzeichnung von Dokumenten für Anlagen, Systeme und Ausrüstungen – Teil 1. Berlin: Beuth 2009.

DIN 62079

DIN 62079: Erstellung von Anleitungen – Gliederung, Inhalt und Darstellung. Berlin: Beuth 2001.

DIN 62264

DIN 62264: Integration von Unternehmens-EDV und Leitsystemen – Aktivitätsmodelle für das operative Produktionsmanagement – Teil 3. Berlin: Beuth 2007.

DIN 66312

DIN 66312: IRL – Industrial Robot Language (zurückgezogen). Berlin: Beuth 1996.

DIN 8373

DIN 8373: Roboter und Robotikgeräte – Wörterbuch. Berlin: Beuth 2010.

DIN 9409

DIN 9409: Industrieroboter – Mechanische Schnittstellen. Berlin: Beuth 2004.

DIN-Fachbericht 62390

DIN-Fachbericht 62390: Leitfaden für Geräteprofile in der Automatisierungstechnik – Deutsche Fassung. Berlin: Beuth 2009.

Draht 2010

Draht, R.: Datenaustausch in der Anlagenplanung mit AutomationML. Heidelberg: Springer 2010.

Dubbel 2007

Dubbel, H.: Taschenbuch für den Maschinenbau. 22. Auflage. Berlin: Springer 2007.

Ebel et al. 2007

Ebel, M.; Baumann, M.; Okon, M.: ProduFlexil: Flexible Produktion mit SOA-Architektur und Plug-and-Work-Mechanismus. In: Spath, D.; Weisbecker, A.; Höß, O.; Drawehn, J. (Hrsg.): Tagungsband des Stuttgarter Softwaretechnik Forum 2007. Stuttgart, 23. November 2007. Stuttgart: Fraunhofer IRB Verlag 2007. S. 65–74.

Epple 2003

Epple, U.: Austausch von Anlagenplanungsdaten auf Grundlage von Metamodellen. atp Automatisierungstechnische Praxis 45 (2003) 7, S. 61–70.

EPSG 2008

EPSG: Ethernet POWERLINK Communication Profile Specification. Ethernet Powerlink Standardization Group (EPSG). Berlin 2008.

Feldmann et al. 2007a

Feldmann, K.; Weber, M.; Wolf, W.: Design of a theoretical holistic system model as base of construction kits for building Plug&Produce-able modular production systems. Production Engineering 1 (2007) 3, S. 329–336.

Feldmann et al. 2007b

Feldmann, K.; Wolf, W.; Weber, M.: Design of a formal model for the specification of agent platforms based on Plug&Produce-able production systems. Production Engineering 1 (2007) 3, S. 321–328.

Freisler 2008

Freisler, S.: Technische Dokumentation im Maschinen- und Anlagenbau. <http://www.contentmanager.de/magazin/technische_dokumentation_im_maschinen_und_anlagenbau.html> – 17.03.2012.

Furrer 2000

Furrer, F. J.: Ethernet-TCP/IP für die Industrieautomation – Grundlagen und Praxis. 2. Auflage. Heidelberg: Hüthig 2000.

Fussel 2000

Fussel, B.: Intelligente Feldgeräte und ihre Integration in Produktionssysteme. Dissertation RWTH Aachen (2000). Aachen: Shaker 2000. (WZL/RWTH Aachen – Berichte aus der Produktionstechnik 21).

Gauss et al. 2005

Gauß, M.; Middelmann, R.; Som, F.: XML-basiertes Kommunikationsprotokoll für Industrieroboter und prozessorgestützte Peripheriegeräte – Einführungsblatt. Frankfurt/Main: VDMA 2005.

Gauss et al. 2006

Gauss, M.; Dai, F.; Som, F.; Zimmermann, U. E.; Woern, H.: A Standard Communication Interface for Industrial Robots and Processor Based Peripherals - XIRP. In: VDI-Wissensforum GmbH, D. (Hrsg.): Proceedings for the joint conference of ISR 2006 (37^{st} International Symposium on Robotics) and ROBOTIK 2006 (4^{th} German Conference on Robotics). München, 15.-17. Mai 2006. München: VDI-Verlag 2006. Erschienen als CD.

Grunwald et al. 2008

Grunwald, G.; Plank, G.; Reintsema, D.; Zimmermann, U.: Communication, Configuration, Application – Three Layer Concept for Plug-and-Produce. In: Filipe, J. (Hrsg.): Proceedings of the 5^{th} International Conference on Informatics in Control, Automation and Robotics (ICINCO 2008). Funchal Madeira (Portugal), 11.-15. Mai 2008. Setúbal (Portugal): INSTICC Press 2008. S. 255–262.

Hägele et al. 2005
Hägele, M.; Skordas, T.; Sagert, S.; Bischoff, R.; Brogardh, T.; Dresselhaus, M.: White paper - Industrial Robot Automation: EURON - European Robotics Network, 2005.

Halsall 1996
Halsall, F.: Data Communications, Computer Networks and Open Systems. 4. Auflage. Wokingham (England): Addison-Wesley 1996.

Harbach et al. 2007
Harbach, F.; Janschek, K.; Jumar, U.; Sawodny, O.; Spohr, G.-U.: Herausfordernde Anwendungsgebiete der Automatisierungstechnik. at – Automatisierungstechnik 55 (2007) 5, S. 260–265.

Hedelind & Jackson 2007
Hedelind, M.; Jackson, M.: The Need for Reconfigurable Robotic Systems. In: ElMaraghy, H.; Zäh, M. F. (Hrsg.): Proceedings of the 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2007), Toronto (Kanada), 22.-24. Juli 2007. S. 1253–1262.

Hesse & Malisa 2010
Hesse, S.; Malisa, V. (Hrsg.): Taschenbuch Robotik – Montage – Handhabung. München: Hanser 2010.

Hesse & Schnell 2004
Hesse, S.; Schnell, G.: Sensoren in der Automatisierungstechnik. 3. Auflage. Wiesbaden: Vieweg 2004.

Hill 2009
Hill, D. (Hrsg.): Microsoft Application Architecture Guide – patterns & practices. 2. Auflage. Microsoft Corporation 2009.

Hulzebosch 2008
Hulzebosch, J.: USB in der Elektronik – Die USB-Schnittstelle für praktische Anwendungen am PC einsetzen. Poing: Franzis 2008.

Humburger 1998
Humburger, R.: Konzeption eines Systems zur aufgabenorientierten Roboterprogrammierung. Dissertation RWTH Aachen (1998). Aachen: Shaker 1998. (WZL/RWTH Aachen – Berichte aus der Produktionstechnik 4).

Husty et al. 1997
Husty, M.; Karger, A.; Sachs, H.; Steinhilper, W.: Kinematik und Robotik. Berlin: Springer 1997.

IEC 61784
IEC 61784: Industrial Communication Networks – Profiles – Part 2: Additional Fieldbus Profiles for Real-Time Networks Based on ISO/IEC 8802-3. Berlin-Offenbach: VDE-Verlag 2010.

IEC 62264
IEC 62264: Enterprise-Control System Integration. Berlin-Offenbach: VDE-Verlag 2003.

IEC 62424
IEC 62424: Darstellung von Aufgaben der Prozessleittechnik – Fließbilder und Datenaustausch zwischen EDV-Werkzeugen zur Fließbilderstellung und CAE-Systemen. Berlin-Offenbach: VDE-Verlag 2010.

IFR 2011
IFR: World Robotics 2011 – Industrial Robots. Frankfurt: International Federation of Robotics (IFR) 2011.

I*PROMS 2006
I*PROMS: Innovative Production Machines and Systems - Updated State-of-the-art review of all PAC Cluster research areas, University of Naples Frederico II (UNINA), 2006.

ISO/IEC 7498
ISO/IEC 7498: Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. Berlin-Offenbach: VDE-Verlag 1994.

Jasperneite 2005
Jasperneite, J.: Echtzeit-Ethernet im Überblick. atp Automatisierungstechnische Praxis 47 (2005) 3, S. 29–34.

Jensen et al. 2009
Jensen, S.; Pleßow, M.; Kolster, D.; Herfs, W.: Projekt EmsA – Erstellung eines Entwicklungssystems für modulare, selbstkonfigurierende Visualisierungen zur Anlagenüberwachung. EmsA Schlussbericht, WZL – RWTH Aachen, 2009.

Jäger 2009
Jäger, E.: Industrial Ethernet – Funktionsweise, Implementierung und Programmierung von Feldgeräten mit netX. Heidelberg: Hüthig 2009.

JRA 2008
JRA: ORiN 2.1 Specifications - Version 2.1.0. Tokyo (Japan): Japan Robot Associaton (JRA) 2008.

Kather & Voigt 2008
Kather, A.; Voigt, T.: Vertikale Integration von Getränkeabfüllanlagen auf Basis standardisierter Schnittstellen. PPS Management 13 (2008) 1, S. 40–43.

Kauffels 1996
Kauffels, F.-J.: Einführung in die Datenkommunikation – Grundlagen – Systeme – Dienste. 5. Auflage. Bergheim: Datacom 1996.

Keibel 2003
Keibel, A.: Konzeption und Realisierung eines integrierten Moduls zur Simulation und Steuerung von Kinematiksystemen. Dissertation Universität Dortmund (2003).

Kerner 1995
Kerner, H.: Rechnernetze nach OSI. 3. Auflage. Bonn: Addison-Wesley 1995.

Kuppinger 2004
Kuppinger, S.: Ist Ethernet marktreif? IEE – Elektrische Automatisierung + Antriebstechnik 49 (2004) 12, S. 40–43.

Lambrecht et al. 2011
Lambrecht, J.; Kleinsorge, M.; Krüger, J.: Markerless Gesture-Based Motion Control and Programming of Industrial Robots. In: Proceedings of the 16th International IEEE Conference on Emerging Technologies & Factory Automation (ETFA 2011). Toulouse (Frankreich), 05.-09. September 2011. Erschienen online.

Langmann 2010
Langmann, R.: Taschenbuch der Automatisierung. 2. Auflage. München: Carl Hanser 2010.

Lüder 2005
Lüder, A. (Hrsg.): IAONA Handbook - Industrial Ethernet. 3. Auflage. Magdeburg 2005.

Malec et al. 2007
Malec, J.; Nilsson, A.; Nilsson, K.; Nowaczyk, S.: Knowledge-Based Reconfiguration of Automation Systems. In: Proceedings of the 3rd IEEE International Conference on Automation Science and Engineering (CASE 2007). Scottsdale / Arizona (USA), 22.-25. September 2007 2007, S. 170–175.

Maschinenrichtlinie 2006
Maschinenrichtlinie: Richtlinie 2006/42/EG des europäischen Parlaments und des Rates vom 17. Mai 2006 über Maschinen und zur Änderung der Richtlinie 95/16/EG (Neufassung). Europäische Union 2006.

Mayr & Draht 2007
Mayr, G.; Draht, R.: IEC PAS 62424 – Grafische Darstellung PLT-Aufgaben und Datenaustausch zu Engineering-Systemen. atp Automatisierungstechnische Praxis 49 (2007) 5, S. 22–29.

Meyer 2002
Meyer, M.: Kommunikationstechnik. 2. Auflage. Braunschweig: Vieweg 2002.

Milberg 1992
Milberg, J. (Hrsg.): Von CAD, CAM zu CIM. Berlin: Springer 1992.

Mizukawa et al. 2002
Mizukawa, M.; Matsuka, H.; Koyama, T.; Inukai, T.; Noda, A.; Tezuka, H.; Noguchi, Y.; Otera, N.: ORiN – Open Robot Interface for the Network. In: Proceedings of the 41st Annual SICE Conference 2002. Osaka (Japan), 05.-07. August 2002. Tokyo (Japan): Society of Instrument and Control Engineers 2002. S. 925–928.

Mizukawa et al. 2004
Mizukawa, M.; Sakakibara, S.; Otera, N.: Implementation and Applications of Open Data Network Interface 'ORiN'. In: Proceedings of the 43rd Annual SICE Conference 2004. Okayama (Japan), 08.-10. Oktober 2004. Tokyo (Japan): Society of Instrument and Control Engineers 2004, S. 1340–1343.

Naumann et al. 2006

Naumann, M.; Wegener, K.; Schraft, R. D.: Robot Cell Integration by Means of Application-P'n'P. In: VDI-Wissensforum; DGR (Hrsg.): Proceedings for the joint conference of ISR 2006 (37^{st} International Symposium on Robotics) and ROBOTIK 2006 (4^{th} German Conference on Robotics). München, 15.-17. Mai 2006. München: VDI-Verlag 2006. Erschienen als CD.

Naumann et al. 2007

Naumann, M.; Wegener, K.; Schraft, R. D.: Control Architecture for Robot Cells to Enable Plug'n'Produce. In: Hutchinson, S. (Hrsg.): IEEE International Conference on Robotics and Automation (ICRA 2007), Rom (Italien), 10.-14. April 2007. S. 287–292.

Neugebauer 1997

Neugebauer, J.-G.: Einsatz neuer Mensch-Maschine-Schnittstellen für Robotersimulation und -programmierung. Dissertation Universität Stuttgart (1997). Berlin: Springer 1997. (IPA-IAO – Forschung und Praxis 256).

Nof 1999

Nof, S. Y. (Hrsg.): Handbook of Industrial Robotics. 2. Auflage. New York (USA): Wiley 1999.

Nyhuis et al. 2008

Nyhuis, P.; Reinhart, G.; Abele, P. (Hrsg.): Wandlungsfähige Produktionssysteme – Heute die Industrie von morgen gestalten. Hannover: PZH-Verlag 2008.

OSACA 1996

OSACA: Open System Architecture for Controls within Automation Systems. OSACA I + II Schlussbericht, WZL – RWTH Aachen, 1996.

Plank et al. 2006

Plank, G.; Reintsema, D.; Grunwald, G.; Otter, M.; Kurze, M.; Löhning, M.; Reiner, M.; Zimmermann, U.; Schreiber, G.; Weiss, M.; Bischoff, R.; Fellhauer, B.; Notheis, T.; Barklage, T.: PAPAS - Plug and Play Antriebs- und Steuerungskonzepte für die Produktion von Morgen: Forschung für die Produktion von morgen Schlüsselkomponente Handhabungstechnik.

PAPAS Abschlussbericht, Deutsches Zentrum für Luft- und Raumfahrt (DLR) et. al., 2006.

Pritschow 1997
Pritschow, G.: HÜMNOS - Trendwende in der Steuerungstechnik. Kurzbericht im Rahmen der EMO (Werkzeugmaschinen-Weltausstellung) 1997, ISW – Universität Stuttgart, 1997.

Pöschmann & Krogel 2000
Pöschmann, A.; Krogel, P.: Autoconfiguration Management für Feldbusse – PROFIBUS Plug & Play. e&i Elektrotechnik und Informationstechnik 117 (2000) 5, S. 335–339.

Reinhart & Krug 2009
Reinhart, G.; Krug, S.: Robotersysteme in der Kleinserie - Effizient von der Planung bis zum Einsatz, München: Utz 2009, (*iwb* Seminarberichte 91). S. 1–16.

Reinhart & Krug 2010
Reinhart, G.; Krug, S.: Current State Model for Easy Reconfiguration of Robot Systems and Offline-Programming-Environments. In: ITG; VDMA; IFR; DGR (Hrsg.): Proceedings for the joint conference of ISR 2010 (41^{st} International Symposium on Robotics) and ROBOTIK 2010 (6^{th} German Conference on Robotics). München, 07.-09. Juni 2010. Berlin-Offenbach: VDE-Verlag 2010. Erschienen als CD.

Reinhart et al. 2010
Reinhart, G.; Krug, S.; Hüttner, S.; Mari, Z.; Riedelbauch, F.; Schlögel, M.: Automatic Configuration (Plug&Produce) of Industrial Ethernet Networks. In: Cardoso, J. R. (Hrsg.): Proceedings of the 9^{th} IEEE/IAS International Conference on Industry Applications (INDUSCON 2010), São Paulo (Brasilien), 08.-10. November 2010. Erschienen online <http://www.ieee.org.br/induscon/2010>.

Reinhart et al. 2011
Reinhart, G.; Hüttner, S.; Krug, S.: Automatic Configuration of Robot Systems – Upward and Downward Integration. In: Jeschke, S.; Liu, H.; Schilberg, D. (Hrsg.): Proceedings of the 4^{th} International Conference

on Intelligent Robotics and Applications (ICIRA 2011). Aachen, 06.-08. Dezember 2011. Heidelberg: Springer 2011. S. 102–111.

Ropohl 2009
Ropohl, G.: Allgemeine Technologie – Eine Systemtheorie der Technik. 3. Auflage. Karlsruhe: Universitätsverlag Karlsruhe 2009.

Sagert 2006
Sagert, S.: Neue VDMA-Einheitsblätter. VDMA Nachrichten (2006) 11, S. 30.

Sauer 2008a
Sauer, O.: „Plug and work“ funktioniert am Band – Produktionsnahe IT-Systeme werden intelligenter und komfortabler. Intelligenter Produzieren (2008) 5, S. 8–10.

Sauer 2008b
Sauer, O.: Automated engineering of manufacturing execution systems – a contribution to „adaptivity“ in manufacturing companies. In: Bernard, A. (Hrsg.): Proceedings of the 5th International Conference on Digital Enterprise Technology (DET 2008), Nantes (Frankreich), 22.-24. Oktober 2008. S. 181–191.

Schaffeld 2011
Schaffeld, D.: Ethernet Powerlink – Daten in Echtzeit mit Ethernet – Glossar. <http://et.fh-duesseldorf.de/home/langmann/seminarprojekte/Moodle/ethernetPowerlink/glossar.html> – 12.12.2011, 2011.

Schleipen et al. 2009
Schleipen, M.; Okon, M.; Baumann, M.; Neukäufer, M.; Fedrowitz, C.; Feike, M.; Popova, N.; Nick, M.; Schneickert, S.; Wessner, M.: Design and engineering processes in highly adaptive plants with ambient intelligence techniques. In: Tichkiewitch, S.; Brissaud, D.; Frein, Y. (Hrsg.): Proceedings of 42nd CIRP Conference on Manufacturing Systems – Sustainable Development of Manufacturing Systems, Grenoble (Frankreich), 03.-05. Juni 2009. Erschienen als CD.

Schnell 2006
Schnell, G. (Hrsg.): Bussysteme in der Automatisierungs- und Prozesstechnik – Grundlagen, Systeme und Trends der industriellen Kommunikation. 6. Auflage. Wiesbaden: Vieweg 2006.

Schwager 2004a
Schwager, J.: Ethernet erreicht das Feld: Sechs Echtzeit-Variantan im Vergleich – Teil 1. Elektronik 86 (2004) 11, S. 48–54.

Schwager 2004b
Schwager, J.: Ethernet erreicht das Feld: Sechs Echtzeit-Varianten im Vergleich – Teil 2. Elektronik 86 (2004) 13, S. 38–43.

Schwarzkopf 2006
Schwarzkopf, P.: Im Fokus Bildverarbeitung: Nimm drei. VDMA Nachrichten (2006) 9, S. 52–57.

Siciliano & Khatib 2008
Siciliano, B.; Khatib, O. (Hrsg.): Handbook of Robotics. Berlin, Heidelberg: Springer 2008.

Siegert & Bocionek 1996
Siegert, H.-J.; Bocionek, S.: Robotik – Programmierung intelligenter Roboter. Berlin: Springer 1996.

Spath & Scholtz 2007
Spath, D.; Scholtz, O.: Ideen gegen Verlagerung der Montage ins Ausland. wt Werkstattstechnik online 97 (2007) 1/2, S. 2–7.

Stallings 2004
Stallings, W.: Data and Computer Communications. 7. Auflage. New Jersey (USA): Pearson 2004.

UPnP 2008
UPnP: UPnP Device Architecture 1.0. Online: 2008. <http://www.heise.de/netze/artikel/Netzwerke-mit-UPnP-einrichten-und-steuern-221520.html> - 28.12.2011.

VDI-Richtlinie 2860
VDI-Richtlinie 2860: Montage- und Handhabungstechnik – Handhabungsfunktionen, Handhabungseinrichtungen, Begriffe, Definitionen, Symbole. Berlin: Beuth 1990.

VDI-Richtlinie 2863
VDI-Richtlinie 2863: Programmierung numerisch gesteuerter Handhabungseinrichtungen. Berlin: Beuth 1987.

VDMA 66430
VDMA 66430: XML-basiertes Kommunikationsprotokoll für Industrieroboter und prozessorgestützte Peripheriegeräte (XIRP). Berlin: Beuth 2006.

Verl & Naumann 2008a
Verl, A.; Naumann, M.: Kleine Losgrößen im Griff - Flexible Steuerungsarchitekturen für Automatisierungssysteme. Elektronik 90 (2008) 7/8, S. 64–68.

Verl & Naumann 2008b
Verl, A.; Naumann, M.: Plug'n'Produce-Steuerungsarchitektur für Roboterzellen. wt Werkstattstechnik online 98 (2008) 5, S. 384–390.

Vogl 2008
Vogl, W.: Eine interaktive räumliche Benutzerschnittstelle für die Programmierung von Industrierobotern. Dissertation Techniche Universität München (2008). München: Utz 2008. (Forschungsberichte *iwb* 228).

Voigt & Kather 2005
Voigt, T.; Kather, A.: Weihenstephaner Standards für BDE-Systeme. Getränkeindustrie 59 (2005) 9, S. 158–161.

Weber 2007
Weber, M.: Unterstützung der Wandlungsfähigkeit von Produktionsanlagen durch innovative Softwaresysteme. Dissertation Universität Erlangen-Nürnberg (2007). Bamberg: Meisenbach 2007. (Fertigungstechnik – Erlangen 188).

Weber 2009
Weber, W.: Industrieroboter – Methoden der Steuerung und Regelung. 2. Auflage. München: Carl-Hanser 2009.

Weck 2006

Weck, M.: Werkzeugmaschinen – Automatisierung von Maschinen und Anlagen. 6. Auflage. Heidelberg: Springer 2006.

Wellenreuther & Zastrow 2009

Wellenreuther, G.; Zastrow, D.: Automatisieren mit SPS – Theorie und Praxis. 4. Auflage. Wiesbaden: Vieweg + Teubner 2009.

Zäh et al. 2004

Zäh, M. F.; Vogl, W.; Munzert, U.: Beschleunigte Programmierung von Industrierobotern – Augmented Reality-Einsatz zur intuitiven Mensch-Maschine-Interaktion. wt Werkstattstechnik online 94 (2004) 9, S. 438–441.

Verzeichnis betreuter Studienarbeiten

Im Rahmen dieser Dissertation entstanden am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) in den Jahren 2009 bis 2012 unter wesentlicher wissenschaftlicher, fachlicher und inhaltlicher Anleitung des Autors, die im Folgenden aufgeführten studentischen Arbeiten. In diesen Arbeiten wurden verschiedene Fragestellungen zur automatischen Konfiguration von Robotersystemen untersucht, deren Ergebnisse in Teilen in das vorliegende Dokument eingeflossen sind. Der Autor dankt allen Studierenden für ihr Engagement bei der Unterstützung dieser wissenschaftlichen Arbeit.

Bauer 2012

Bauer, Stefan: Integration der Offline-Programmierung in Plug&Produce-Robotersysteme. Semesterarbeit (Nr. 2011/002-S). Bearbeitungszeitraum: 10.07.2011 – 01.01.2012.

Braun 2011

Braun, Lukas: Methode zur Plug&Produce-Integration eines sich ändernden Moduls in ein Robotersystem am Beispiel eines Schweißgerätes und eines Echtzeithubs. Bachelor's Thesis (Nr. 2011/033-B). Bearbeitungszeitraum: 01.05.2011 – 01.11.2011.

Hammerstingl 2012

Hammerstingl, Veit: Konzeption eines Konfigurationsmanagers für Plug&Produce-Robotersysteme. Diplomarbeit (Nr. 2011/046-D). Bearbeitungszeitraum: 01.01.2012 – 01.07.2012.

Mari 2010

Mari, Zeyad: Entwicklung und Intergration einer selbstkonfigurierenden Industrial-Ethernet-Schnittstelle bei Industrierobotern am Beispiel eines Schweißwerkzeugs. Diplomarbeit (Nr. 2009/53). Bearbeitungszeitraum: 14.12.2009 – 14.06.2010.

Schlögel 2010

Schlögel, Mark: Erstellung einer selbstkonfigurierenden Schnittstelle für die automatische Anbindung eines Dreh-Kipp-Tisches an einen Industrieroboter. Master's Thesis (Nr. 2010/1). Bearbeitungszeitraum: 01.02.2010 – 01.08.2010

Schmidt 2010

Schmidt, Thomas W.: Entwicklung eines Postprozessors für die automatische Konfiguration eines Industrieroboters. Semesterarbeit (Nr. 2009/12). Bearbeitungszeitraum: 15.02.2010 – 15.08.2010.

Anhang

In diesem Anhang werden unter anderem informationstechnische Aspekte dargestellt, die teilweise etablierte informationstechnische Strukturen und Ausdrücke in englischer Sprache (z. B. AutomationML) enthalten. Zur einheitlichen Darstellung sind die betreffenden Abbildungen ausschließlich in Englisch beschrieben.

Anhang A: Zustandsmodell

Abbildung 83 zeigt die Struktur des Zustandsmodells, umgesetzt in AutomationML. Die Darstellung erfolgte mithilfe des AutomationML-nativen Editors, welcher das XML-Format grafisch aufbereitet.

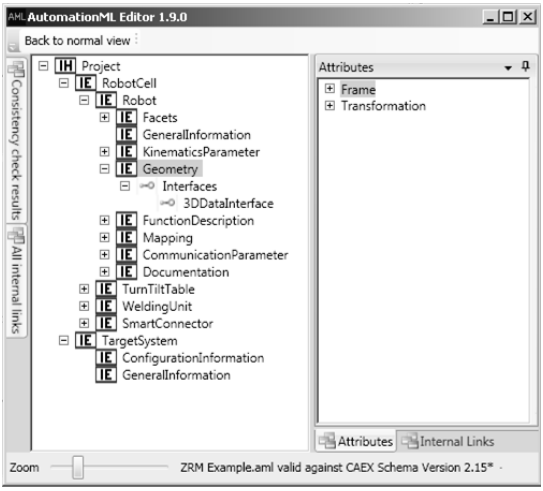

Abbildung 83: Ansicht des Zustandsmodells im AutomationML-Editor

In Abbildung 84 ist eine Übersicht der Struktur des Zustandsmodells dargestellt. Die darauffolgenden Abbildungen 85-89 beschreiben die einzelnen Elemente im Detail.

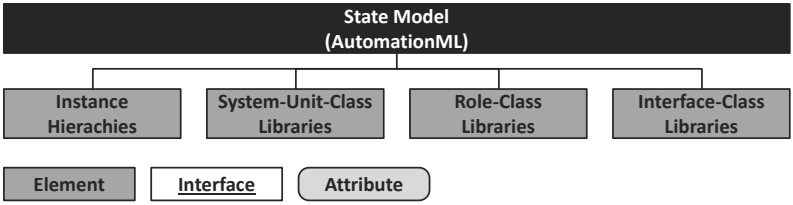

Abbildung 84: Strukturübersicht des Zustandsmodells

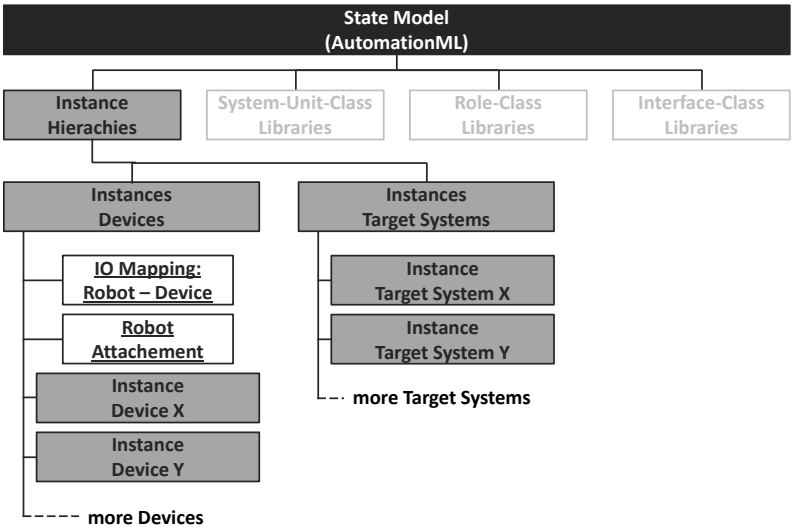

Abbildung 85: Struktur des Zustandsmodells Teil 1

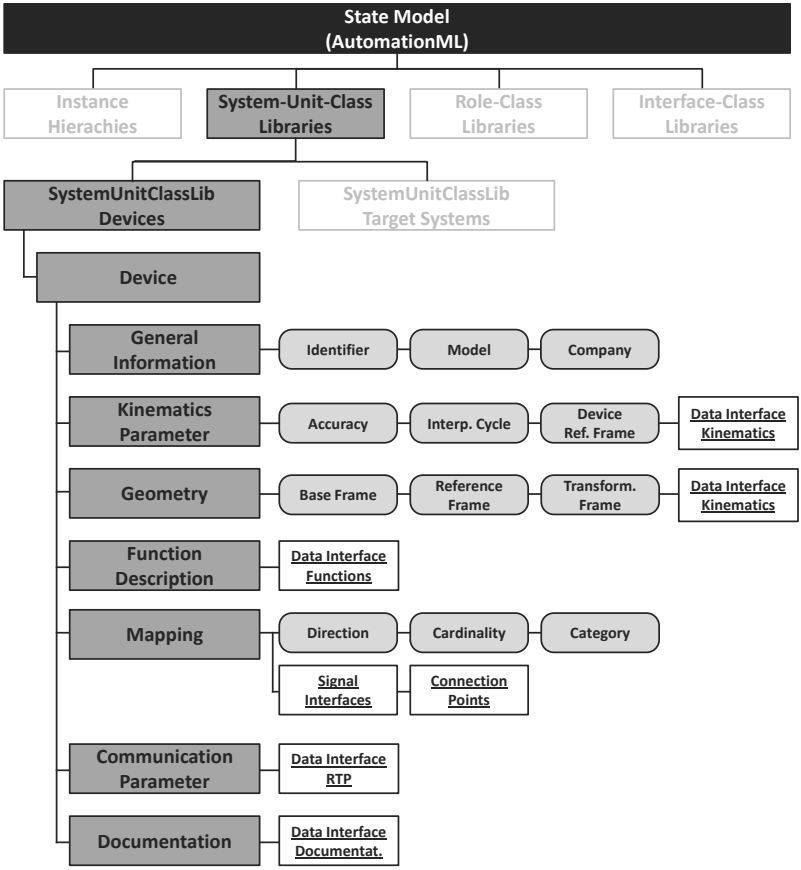

Abbildung 86: Struktur des Zustandsmodells Teil 2

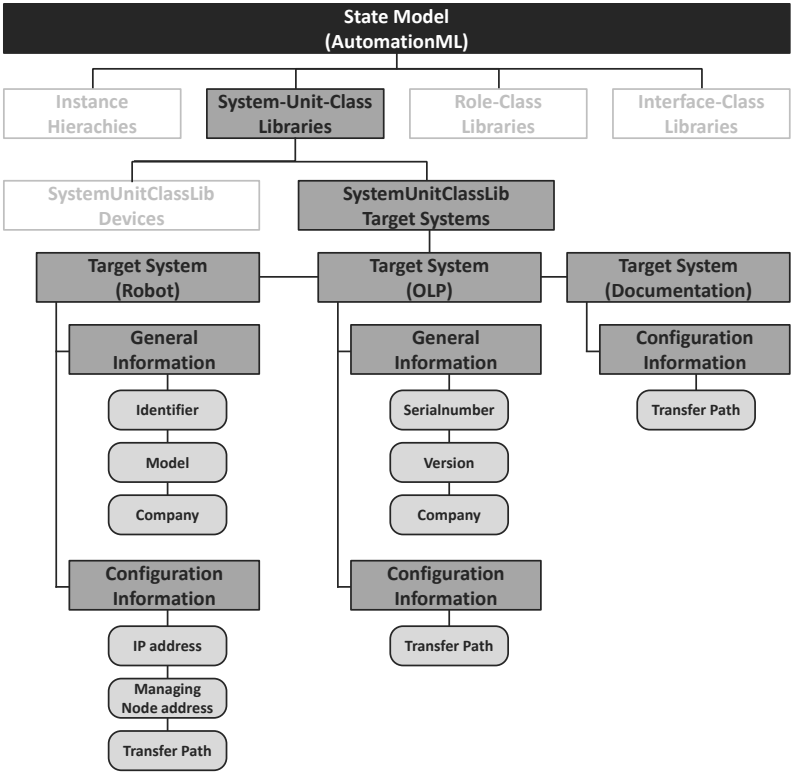

Abbildung 87: Struktur des Zustandsmodells Teil 3

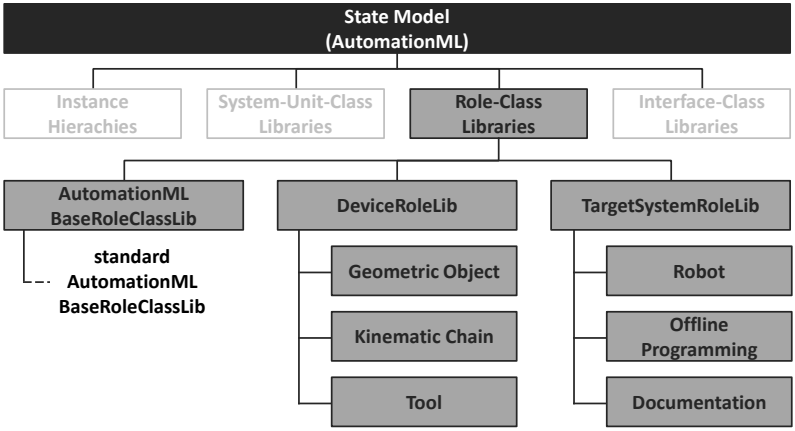

Abbildung 88: Struktur des Zustandsmodells Teil 4

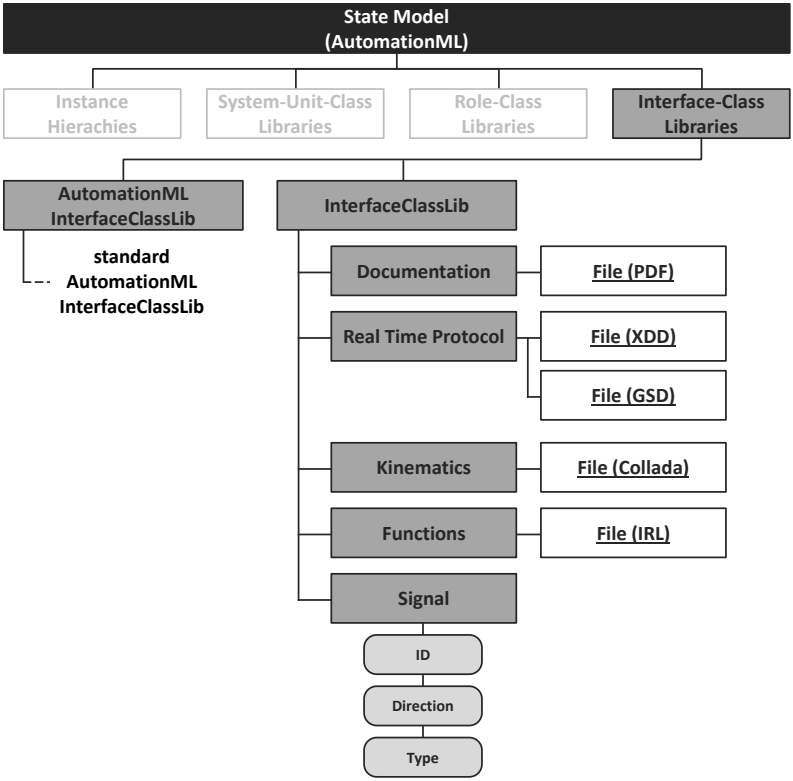

Abbildung 89: Struktur des Zustandsmodells Teil 5

Anhang B: Gerätebeschreibung

Abbindung 90 zeigt beispielhaft die Struktur der Gerätebeschreibung, welche in dem dargestellten Testaufbau für den Dreh-Kipp-Positionierer eingesetzt wurde.

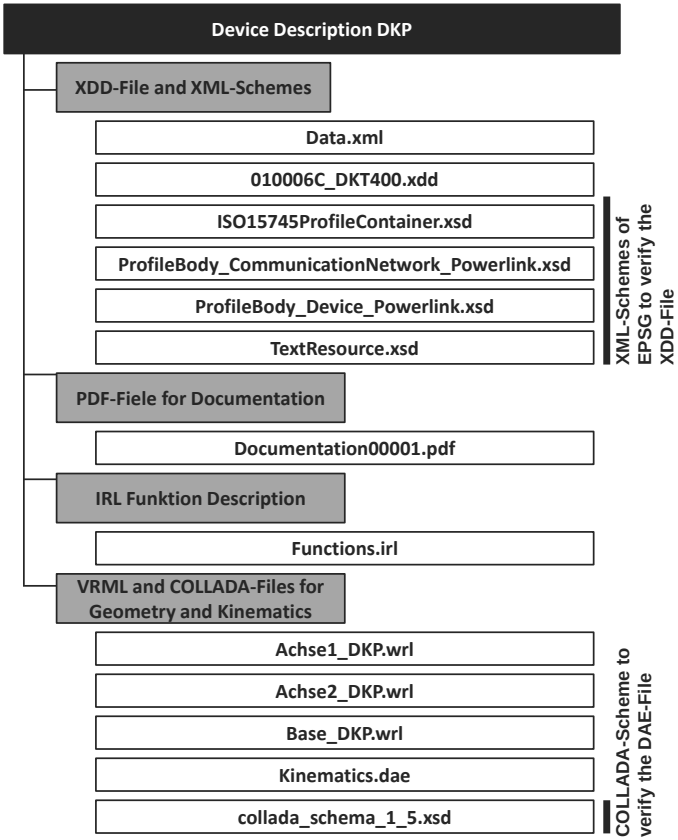

Abbildung 90: Übersicht über die Bestandteile der Gerätebeschreibung des eingesetzten Dreh-Kipp-Positionierers

Die verwendeten Standardformate XDD, PDF, IRL, WRL und DAE wurden unverändert angewendet. Exemplarisch zeigt Abbildung 91 einen Teil der XDD-Datei, welche die Informationen zur Prozessdatenzuordnung enthält.

```
<!-- movement variables -->
<Object index="A580" name="UINT_Input" objectType="9">
    <SubObject subIndex="00" name="NumberOfEntries" objectType="7" dataType="0005"
      accessType="const" PDOmapping="no" defaultValue="6"/>
    <SubObject subIndex="01" name="Axis7setPos" objectType="7" dataType="0006"
      accessType="wo" PDOmapping="optional"/>
    <SubObject subIndex="02" name="Axis7setVel" objectType="7" dataType="0006"
      accessType="wo" PDOmapping="optional"/>
    <SubObject subIndex="03" name="Axis8setPos" objectType="7" dataType="0006"
      accessType="wo" PDOmapping="optional"/>
    <SubObject subIndex="04" name="Axis8setVel" objectType="7" dataType="0006"
      accessType="wo" PDOmapping="optional"/>
    <SubObject subIndex="05" name="Axis7setAcc" objectType="7" dataType="0006"
      accessType="wo" PDOmapping="optional"/>
    <SubObject subIndex="06" name="Axis8setAcc" objectType="7" dataType="0006"
      accessType="wo" PDOmapping="optional"/>
</Object>
```

Abbildung 91: Ausschnitt der Prozessdatenzuordnung der XDD-Datei

Anhang C: UML-Sequenzdiagramm

Abbildungen 92 bis 95 zeigen den Programmablauf des Prototypen.

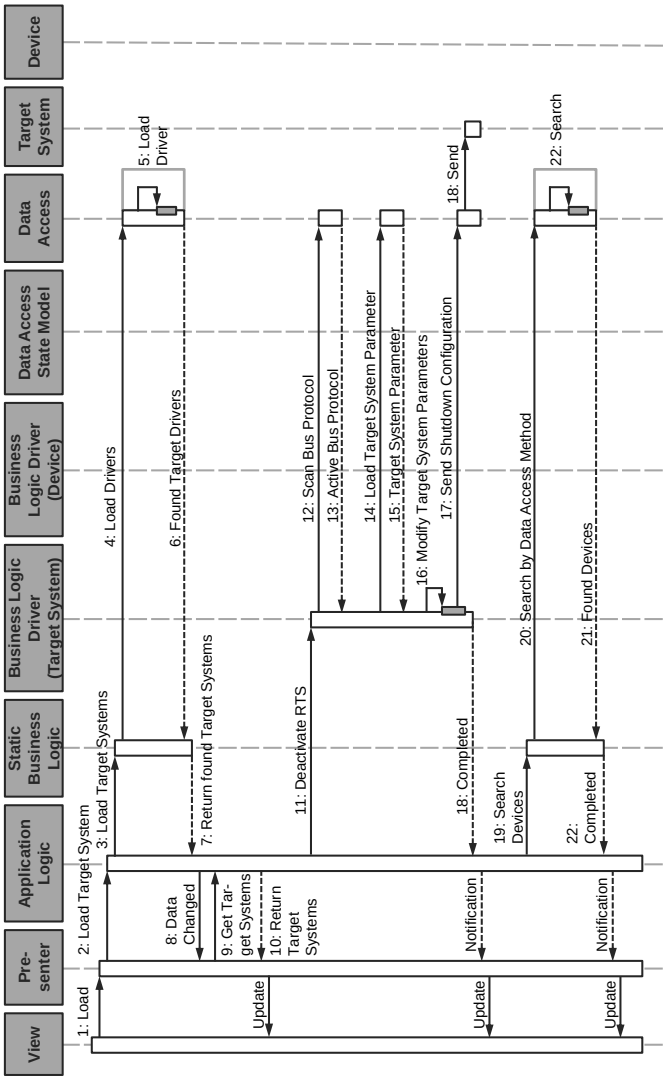

Abbildung 92: Programm-Sequenz Teil 1

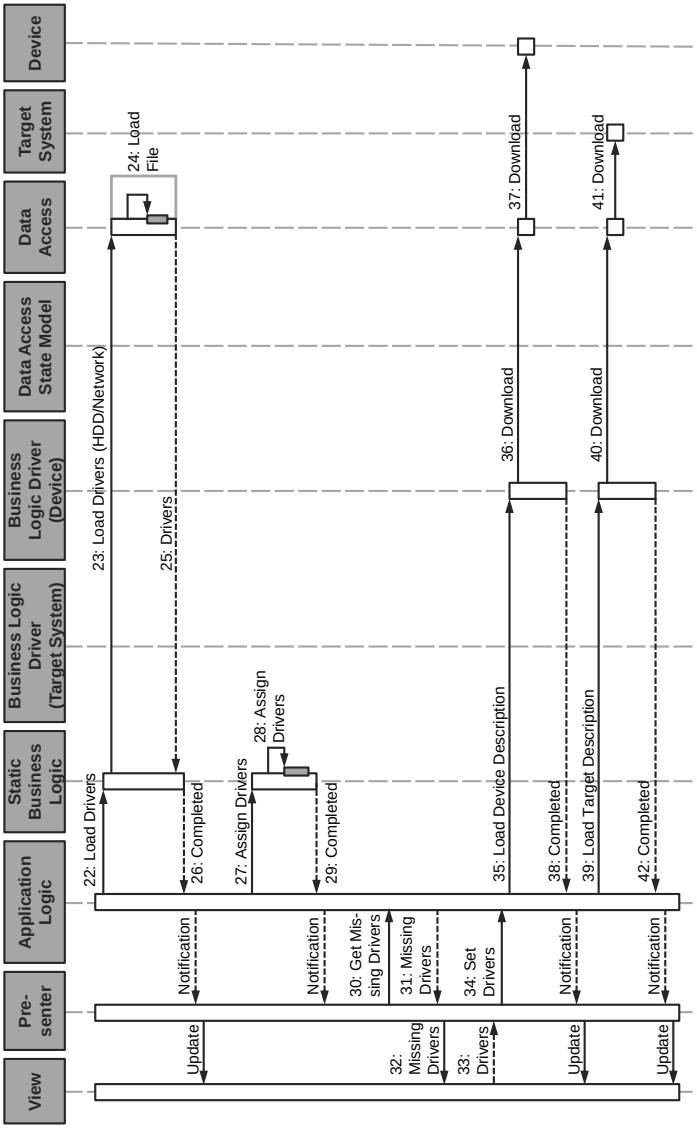

Abbildung 93: Programm-Sequenz Teil 2

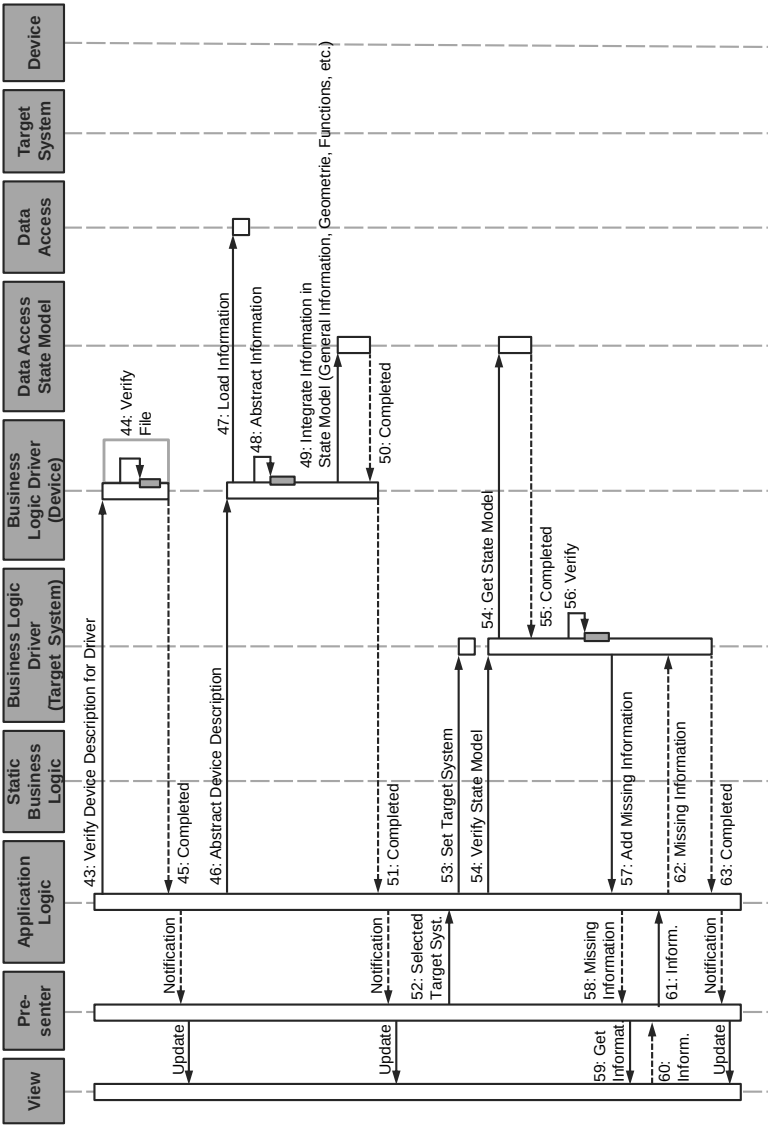

Abbildung 94: Programm-Sequenz Teil 3

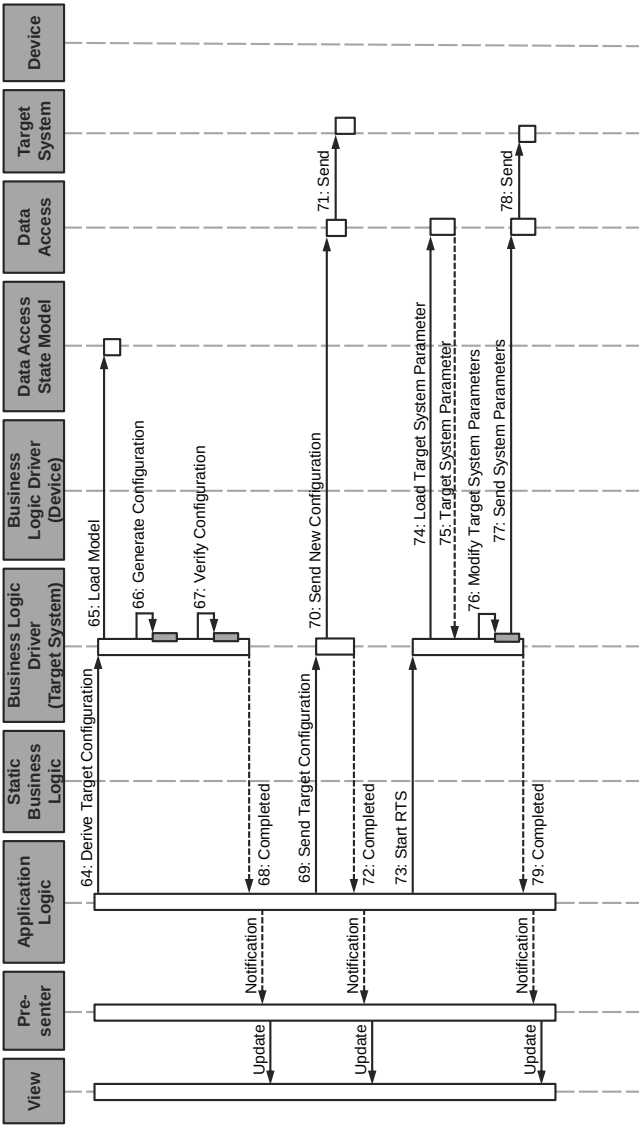

Abbildung 95: Programm-Sequenz Teil 4

Anhang D: Probandenstudie

	mit Erfahrung				ohne Erfahrung			
VP = Versuchsperson	VP 1	VP 2	VP 3	VP 4	VP 5	VP 6	VP 7	VP 8
Selbsteinschätzung 1 = Experte; 5 = Laie	2	1	3	1	5	4	5	5
geschätzte Dauer der konventionellen Konfiguration [h]	30	20	55	10	-	-	-	-
Dauer der Plug&Produce Konfiguration [mm:ss]	07:57	07:45	07:46	08:16	07:35	08:00	07:58	08:37

Tabelle 4: Tabelle der Ergebnisse der Probandenstudie mit der Plug&Produce-Methode für Robotersysteme

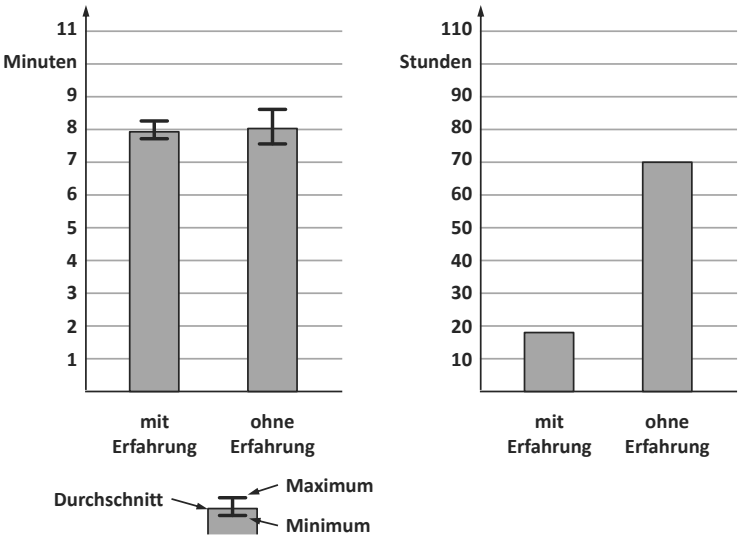

Abbildung 96: Zeiten der acht Versuchspersonen mit der Plug&Produce-Methode (links); Zeiten der zwei Versuchspersonen nach konventionellem Vorgehen (rechts)