TECHNISCHE UNIVERSITÄT MÜNCHEN

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

Adaptierbares aufgabenorientiertes Programmiersystem für Montagesysteme

Julian Christoph Sebastian Backhaus

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. Hon.-Prof. Dr.-Ing. Wilhelm Bauer, Universität Stuttgart

Die Dissertation wurde am 07.07.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 15.12.2015 angenommen.

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 sowie als Gruppenleiter an der Projektgruppe Ressourceneffiziente mechatronische Verarbeitungsmaschinen (RMV) des Fraunhofer IWU.

Mein besonderer Dank gilt Herrn Prof. Dr.-Ing. Gunther Reinhart sowie Herrn Prof. Dr.-Ing. Michael F. Zäh, den Leitern des *iwb*. Bei Herrn Prof. Reinhart bedanke ich mich herzlich für die Betreuung der Arbeit sowie die Diskussionen zur deren Weiterentwicklung. Herrn Prof. Zäh danke ich sehr für die Übernahme des Vorsitzes der Prüfungskommission.

Sehr gefreut habe ich mich über die Übernahme des Koreferats durch Herrn Prof. Dr.-Ing. Wilhelm Bauer, dem Institutsleiter des Fraunhofer IAO und des IAT der Universität Stuttgart. Ich bin dabei sehr dankbar über seine aufmerksame und kritische Durchsicht der Arbeit.

Darüber hinaus bedanke ich mich recht herzlich bei allen Kollegen. Neben der fachlichen Unterstützung hat insbesondere die kollegiale und kreative Atmosphäre dazu beigetragen, dass mir die Arbeit immer Freude bereitet hat. Insbesondere möchte ich mich bei Dr.-Ing. Robert Wiedenmann, Stefan Hüttner, Markus Westermeier, Zeyad Mari, Johannes Schmalz, Peter Stich, Andreas Hees, Thorsten Klein, Christoph Richter, Christoph Berger, Veit Hammerstingl und Joachim Michniewicz bedanken. Einen herzlichen Dank auch an alle Service Center. Ein besonderer Dank gilt außerdem allen Studentinnen und Studenten, die mich bei der Erstellung meiner Arbeit unterstützt haben.

Ein großer Dank gilt meiner Mutter die mich immer gestärkt hat und allen Verwandten und Freunden, welche mich bei der Verbesserung der Arbeit unterstützt haben. Der größte Dank gilt schließlich meiner Frau Christina, die mir den Rücken freigehalten und mir immer gezeigt hat, dass es auch wichtigere Dinge neben der Promotion gibt.

München, im Februar 2016 *Julian Backhaus*

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abkürzung	Bedeutung
2D	zweidimensional
3D	dreidimensional
AG	Aktiengesellschaft
ANSI	American National Standardization Institute
API	Application Programming Interface
AS	Ablaufsprache
AutomationML	Automation Modeling Language
AWL	Anweisungsliste
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAEX	Computer Aided Engineering Exchange
CAM	Computer Aided Manufacturing
COLLADA™	Collaborative Design Activity
CPS	Cyber-Physical Systems
CPU	Central Processing Unit
DEV	Digital Engineering Visualisation
DH	Denavit-Hartenberg
DIN	Deutsches Institut für Normung
DKP	Dreh-Kipp-Positionierer
DM	Engineering Data Management
DOF	Degree of Freedom
EDDL	Electronic Device Description Language

eEPK	Erweiterte ereignisgesteuerte Prozessketten
EN	Europäische Norm
ERP	Enterprise Resource Planning
FBS	Funktionsbausteinsprache
GmbH	Gesellschaft mit beschränkter Haftung
GSD	Generic Station Description
GSDML	Generic Station Description Markup Language
GUI	Graphical User Interface
ID	Identifikationsnummer
IEC	International Electrotechnical Commission
IGES	Initial Graphics Exchange Specification
IML	Intermediate Modelling Layer
IMS/HMS	Intelligent Manufacturing Systems/ Holonic Manufacturing Systems
IR	Industrieroboter
IRL	Industrial Robot Language
ISA	International Federation of the National Standardizing Associations
ISO	International Organiziation for Standardization
JT	Jupiter Tesselation
KOP	Kontaktplan
KOS	Koordinatensystem
KRL	KUKA Robot Language
MathML	Mathematical Markup Language
MC	Motion Control
MES	Manufacturing Execution System

MIG/MAG	Metal Inert Gas/Metal Aktive Gas
NC	Numerical Control
OSI	Open System Interconnection
Pdf	Portable Document Format
PDM	Product Data Management
PLC	Programmable Logic Controller
PLM	Produce Lifecycle Management
PPR	Produkt, Prozess und Ressource
RC	Robot Control
RWTH	Rheinisch-Westfälische Technische Hochschule
SADT	Structured Analysis and Design Technique
SAPIR	Supervised and Adaptive Programming of Industrial Robots
SCADA	Supervisory Control and Data Acquisition
SFB	Sonderforschungsbereich
SFC	Sequential Fuction Chart
SIARAS	Skill-based Inspection and Assembly of Reconfigurable Automation Systems
SOA	Service Orientied Architekture
SPS	Speicherprogrammierbare Steuerung
ST	Strukturierter Text
STEP	Standard for the Exchange of Product Model Data
SysML	Systems Modeling Language
TCP	Tool-Center-Point
UML	Unified Modelling Language

VDI	Verein Deutscher Ingenieure
VDMA	Verband Deutscher Maschinen- und Anlagenbau e.V.
vgl.	vergleiche
VIBN	Virtuelle Inbetriebnahme
VRML	Virtual Reality Modeling Language
XDD	XML-Device-Description
XML	Extensible Markup Language
XROB	Experimental Platform in Robotics

Verzeichnis der Formelzeichen

Variable	Einheit	Bedeutung
l	m	Länge
x, y, z	mm	Koordinaten
z, x‘, z‘‘	°	Orientierung in Euler Winkeln
AB	€	Abschreibungszeitraum
Z		Kalkulatorische Zinsen
S_{MA}	€	Stundensatz Mitarbeiter
I	€	Invest
t_{prog}	h/Jahr	Stunden für klassische Programmierung
d		Einsparung durch aufgabenorientierte Programmierung
E	h/Jahr	Summe jährlicher Einsparungen
W_J	€/Jahr	Wartung jährlich
A_{eng}	€/Jahr	Zusätzliche Aufwände im Engineering
K	€/Jahr	Summe jährlicher Kosten

1 Einleitung

1.1 Aktuelle Herausforderungen im Lebenszyklus von automatisierten Produktionssystemen

Die Automatisierung der Produktion bietet die Möglichkeit, Arbeitsplätze am Standort Deutschland zu halten, die ansonsten in Regionen mit geringeren Lohnkosten verlagert würden (VDI/VDE 2009). Aus diesem Grund wird die Automatisierung auch als wesentlicher Erfolgsfaktor im Bereich der Produktionssysteme gesehen (NAUMANN ET AL. 2010). Aktuelle Studien zeigen, dass der Trend hin zu automatisierten Produktionsanlagen ungebrochen ist. So stieg die Anzahl verkaufter Industrieroboter im Jahr 2013 um 12 % auf 178.132 Stück, was den bisher höchsten dokumentierten Wert darstellt, und auch in Zukunft soll sie weiter steigen (IFR 2014). Dieser Trend gilt jedoch nicht nur für Industrieroboter, sondern allgemein für den Bereich Automatisierungstechnik, wie deren Zuwachsraten von 6 % bis 15 % in Deutschland in den Jahren 2003 bis 2008 zeigen (VDI/VDE 2009).

Der Automatisierungsgrad bei niedrigen Stückzahlen ist jedoch aktuell noch gering. So setzten nach ARMBRUSTER ET AL. (2009) nur 22 % der Unternehmen Industrieroboter in der Kleinserienfertigung und nur 14 % in der Einzelfertigung ein. Aus der Betrachtung des Lebenszyklus eines Produktionssystems, bzw. einer Fabrik gekreuzt mit dem Produktlebenszyklus (vgl. Abbildung 1), können Herausforderungen und Probleme in den einzelnen Phasen benannt werden, die eine weitere Erhöhung des Automatisierungsgrads der Produktion erschweren.

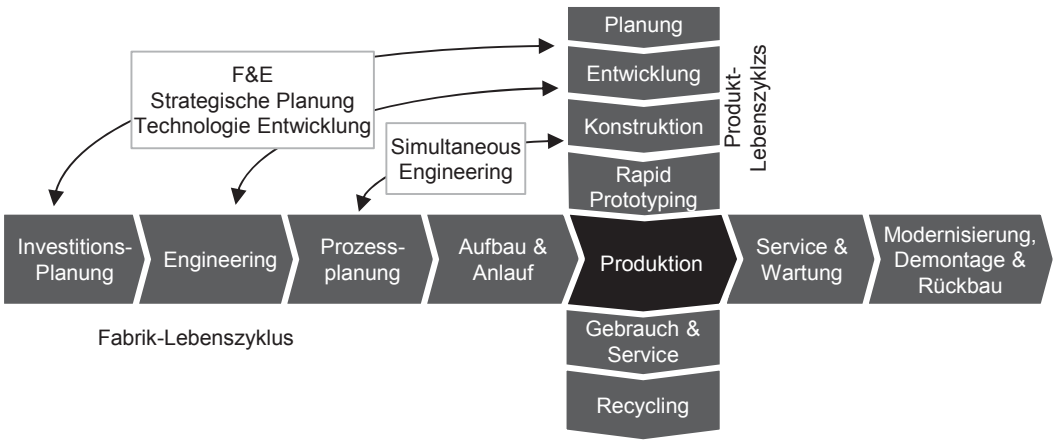


Abbildung 1: Der Produkt- und Fabrik-Lebenszyklus nach WESTKÄMPER (2008)

Diese Herausforderungen lassen sich aus global wirkenden Megatrends ableiten. Insbesondere die zunehmende Individualisierung und die damit verbundene Variantenvielfalt sowie die sich verkürzenden Produktlebenszyklen (ABELE & REINHART 2011) haben großen Einfluss auf den Lebenszyklus automatisierter Produktionssysteme. Häufige Produktänderungen, Schwankungen in der Nachfrage und Änderungen der gesetzlichen Rahmenbedingungen führen zu einer schlechten Prognostizierbarkeit in der *Investitionsplanung* und verlangen auch während der *Produktion* anpassungsfähige Produktionssysteme (KOREN ET AL. 1999, S. 1). Das *Engineering* für die Neuentwicklung oder die Änderung eines Produktionssystems, die Inbetriebnahme und der Anlauf stellen jedoch aktuell ein Problemfeld dar (KOREN ET AL. 1999, S. 1). So benennen HEDELIND & JACKSON (2007) die Reduktion des Aufwands zur Programmierung und Konfiguration als wesentliches Handlungsfelder im Bereich der Robotersysteme, um diese flexibler zu gestalten. Eine anpassungsfähige Montagetechnik wird auch von REINHART (2000) als Erfolgsfaktor angegeben. Vor allem für kleine und mittlere Unternehmen (KMU) stellt die Anpassungsfähigkeit komplexer Produktionssysteme einen wesentlichen Erfolgsfaktor dar (BENGEL 2008). Beispielsweise erfordert die Programmierung eines Robotersystems für kleine bis mittlere Stückzahlen einen so hohen Aufwand, dass dessen Einsatz oftmals nicht rentabel ist (DENKENA ET AL. 2005; SCHRAFT & MEYER 2006). Für den Bereich der *Inbetriebnahme* bietet die Virtuelle Inbetriebnahme (VIBN) mit einem digitalen Modell die Vorteile einer frühzeitigen Detektion und Beseitigung von Fehlern in Steuerung und Konstruktion (WÜNSCH 2008). Insgesamt wird im Spannungsfeld der Megatrends der durchgängige Einsatz digitaler Methoden und Werkzeuge im gesamten Lebenszyklus eines Produktionssystems als Lösungsbaustein gesehen (BRACHT ET AL. 2009, S. 1).

Die genannten Herausforderungen haben insbesondere im Bereich der Montage eine besonders hohe Relevanz, da dort die durch die Megatrends auftretenden Turbulenzen direkte Auswirkungen haben (REINHART & ZÄH 2007; WESTKÄMPER 2006). Auch für automatisierte Montagssysteme stellt die Programmierung ein wesentliches Handlungsfeld dar. Aus diesem Grund werden die Lösungsmöglichkeiten und Herausforderungen für den Bereich der Programmierung von Montagesystemen detaillierter diskutiert. Dem typischen Aufbau eines Montagesystems entsprechend, kann dies die Programmierung von Speicherprogrammierbaren Steuerungen (SPS) und Industrierobotersteuerungen (IR) umfassen. Für Industrieroboter fordern NAUMANN ET AL. (2007) eine Abstraktion der Programmierung, um diese auch Anwendern zu ermöglichen, die kein Expertenwissen bzgl. des Systems und der Programmiersprachen besitzen. Sie schlagen daher

eine prozessorientierte Programmierung vor. Der Nutzer soll nur die Vorgaben des Prozesses beschreiben. Diese Abstraktion der Programmierung wird typischerweise als *implizite* oder *aufgabenorientierte* Programmierung bezeichnet (HAUN 2007, S. 239). Auch GOTTSCHALD (2001, S. 20) und BRECHER ET AL. (2004, S. 306) bewerten die aufgabenorientierte Programmierung, insbesondere für KMUs, als vielversprechendsten Ansatz, um die Einsetzbarkeit von Industrierobotern weiter zu erhöhen. Aktuell existieren jedoch keine umfassenden kommerziellen Systeme zur Programmierung von Industrierobotern auf abstrakter Ebene (PAN ET AL. 2010, S. 621–622). Im Bereich der SPS-Programmierung wird mit der modellbasierten Steuerungsentwicklung (WITSCH & VOGEL-HEUSER 2004) ein vergleichbarer Ansatz verfolgt.

Für eine aufgabenorientierte Programmierung eines Montagesystems sind digitale Modelle der Aufgabe bestehend aus Produkt- und Prozessbeschreibungen, sowie des Montagesystems, mit den Funktionen der vorhandenen Geräte, notwendige Voraussetzungen (SECKNER 2008, S. 25). Gegenwärtig ist eine Modellerstellung in der Montageplanung allerdings noch mit hohem Aufwand verbunden. Der Einsatz von Modellen kann nur dann wirtschaftlich sein, wenn der Aufwand für die Erstellung möglichst gering ist (PATRON 2004, S. 17). Wie bei allen modellbasierten Programmierverfahren ist somit eine Reduzierung des Aufwands zur Modellerstellung anzustreben (VOGL 2009, S. 27). Die Modellierung kann folglich als Hemmnis für den Einsatz der aufgabenorientierten Programmierung gesehen werden. Sogenannte Fähigkeiten bzw. Skills werden in der Forschung als verbindendes Element zwischen dem Produktionsprozess und den Funktionen der Geräte erachtet (MAFFAI ET AL. 2011, S. 1) und können die Modellierung vereinfachen. Sie beschreiben auf abstrakter Ebene die Funktionalitäten, welche eine Komponente anbieten kann. So ist beispielsweise „Bewegen“ ein grundsätzlicher Skill eines Roboters. Sie ermöglichen eine Abstraktion der Gerätefunktionalitäten ohne die Eigenheiten von verschiedenen Geräten kennen zu müssen. Das Konzept der Skills und deren Einbindung ist jedoch noch nicht abschließend definiert (MAFFAI ET AL. 2011, S. 1) und wird neben der Modellierung der Komponenten und der Produktionsprozesse als wichtiger zukünftiger Schritt für den Entwurf einer neutralen und offenen Beschreibungssprache für das Konzept der Skills gesehen (MAFFAI ET AL. 2011, S. 5, 2011). Für deren Verwendung bei der Modellierung von Montagesystemen ist es daher essentiell, zu identifizieren, welche Klassen von Skills existieren, wie diese voneinander abgegrenzt und wie diese modelliert werden können. Um eine Übertragbarkeit zu gewährleisten, sind vor allem quantitative Größen heranzuziehen.

1.2 Zielsetzung und Betrachtungsbereich der Arbeit

Den im vorangegangenen Abschnitt dargestellten Überlegungen folgend, ist die Zielsetzung dieser Arbeit die Entwicklung eines *adaptierbaren, aufgabenorientierten Offline-Programmiersystems* für automatisierte Montagesysteme, welche auf einem *Skill-basierten Informationsmodell* aufbaut. Die Adaptierbarkeit bezieht sich insbesondere auf die unterschiedlichen Randbedingungen in Unternehmen bzgl. der betrachteten Montageprozesse, der verfügbaren digitalen Daten und dem vorhandenen Wissen der Mitarbeiter. Das Programmiersystem soll auf einer Skill-basierten Verknüpfung zwischen der Montageprozess- und der Montagesystembeschreibung aufbauen, welche in einem *Informationsmodell* beschrieben ist. Um den Aufwand zur Modellierung möglichst gering zu halten, soll ein Konzept entwickelt werden, welches unter der Verwendung bestehender Daten eine möglichst aufwandsarme Modellierung des Montagesystems und der Montageprozessbeschreibung ermöglicht. Als Grundlage hierfür sollen die Skills dienen, deren Definition und Modellierung zu präzisieren bzw. zu entwerfen ist. Da Eindeutigkeit und Übertragbarkeit der verwendeten Skills wesentliche Herausforderungen sind, liegt der Fokus auf der Entwicklung und Anwendung eines allgemeingültigen *Vorgehens zur Definition von Skills in Montagesystemen*. Funktionalitäten zur Planung von einzelnen Montagevorgängen, wie die Bahnplanung für einen Industrieroboter, liegen nicht im Fokus dieser Arbeit. Deren Anwendbarkeit unter Verwendung des Skill-basierten Informationsmodells soll am Rande gezeigt werden. Für den erfolgreichen Einsatz von IT-Systemen ist die Integration in die bestehenden Unternehmensprozesse und IT-Werkzeuge entscheidend (NEDBAL 2013, S. 4). Auch für den Anwendungsfall der aufgabenorientierten Programmierung müssen die Unternehmen befähigt werden, ein entsprechendes Programmiersystem in den bestehenden Engineering- bzw. Montageplanungsprozess mit den vorhandenen IT-Werkzeugen zu integrieren. Deswegen ist ein Teilziel dieser Arbeit die Entwicklung eines *Vorgehensmodells zur Einführung der aufgabenorientierten Programmierung in den Engineeringprozess eines Unternehmens*.

Das zu entwickelnden Programmierungssystem und das zugehörige Informationsmodell sollen für derzeit industriell eingesetzte Standardkomponenten und Steuerungsarchitekturen gestaltet werden, um einen direkten Transfer in die industrielle Praxis zu ermöglichen. Jedoch ist auf eine Unabhängigkeit des Konzepts von einzelnen Herstellern oder Geräten zu achten. Auf der anderen Seite soll das Programmiersystem so aufgebaut sein, dass es für zukünftige Entwicklungen, wie die Cloud-Technologie, anwendbar ist. Das Programmiersystem und

die Modellierung sollen für die Programmierung und die Generierung von Steuerungsprogrammen für Speicherprogrammierbare Steuerungen (SPS) und Robotersteuerungen (RC) in Montagesystemen entworfen werden, welche mit Aktoren, Sensoren und Vorrichtungen kommunizieren.

In Montagestationen können nach WIELAND (1995, S. 46) drei wesentliche Aufgaben und Schnittstellen im Betrieb unterschieden werden. Dies sind:

- technologiebezogene Aufgaben, welche die eigentliche Montageaufgabe und deren Ablauf repräsentieren,
- organisatorische Aufgaben, die zur Bedienung oder zur Kommunikation mit Systemen der Unternehmenssteuerung beitragen und
- Bereitstellungsaufgaben zum Vereinzeln oder Lagern von Bauteilen bzw. Baugruppen.

In der Steuerungstechnik des Montagesystems kann die gleiche Einteilung für die Bestandteile des Steuerungsprogramms vorgenommen werden. In dieser Arbeit werden technologiebezogene Aufgaben und Bereitstellungsaufgaben fokussiert. Organisatorische Bestandteile des Steuerungsprogramms werden nicht direkt adressiert. Eine einfache Erweiterbarkeit des Konzeptes soll aber vorgesehen werden. Da auch die kommunikationstechnische Konfiguration einen erheblichen Aufwand beim Engineering eines automatisierten Montagesystems darstellt, wird im Rahmen dieser Arbeit von einem vorkonfigurierten System ausgegangen. Um Daten aus der Konfiguration verwenden zu können, welche auch für die Programmierung relevant sind, sind bestehende modellbasierte Konzepte zu untersuchen. Anschließend ist auf diesen aufzubauen.

In dieser Arbeit wird demnach ein methodischer Ansatz für die Struktur des aufgabenorientierten Programmiersystems, der Beschreibung der Skills sowie dem Vorgehensmodell zur Einführung der aufgabenorientierten Programmierung entwickelt. Die Arbeit ist somit der angewandten Forschung zuzuordnen. Um Aussagen über den Gültigkeitsbereich zu erhalten, werden wesentliche Kernaspekte der Arbeit prototypisch umgesetzt, wobei eine Validierung für alle möglichen Bereiche nicht im Rahmen der Arbeit möglich ist.

1.3 Aufbau der Arbeit

Der Aufbau dieser Arbeit gliedert sich in insgesamt neun Kapitel und ist in Abbildung 2 dargestellt. Nachdem im ersten Kapitel die Motivation sowie die Zielsetzung dieser Arbeit geschildert werden, erfolgen im anschließenden Kapitel eine Begriffsbestimmung und die Darstellung der wesentlichen Grundlagen für das Verständnis dieser Arbeit. Im Kapitel „Stand der Wissenschaft und Technik“ werden bestehende Konzepte und Vorgehensmodelle vorgestellt und mit der Zielsetzung abgeglichen. In der darauffolgenden Anforderungsanalyse werden sowohl aus der Literatur als auch aus der Betrachtung von Anwendungsfällen die detaillierten Anforderungen für diese Arbeit abgeleitet. Im anschließenden fünften Kapitel wird das adaptierbare aufgabenorientierte Programmiersystem entwickelt, wobei der Fokus auf der Beschreibung des Informationsmodells sowie dem Planungsmodul, dem eigentlichen Kern des Programmiersystems, liegt. Das entwickelte Vorgehensmodell zur Integration der aufgabenorientierten Programmierung in den Engineeringprozess eines Unternehmens wird im sechsten Kapitel erläutert. Im siebten Kapitel wird die Umsetzung und Erprobung der Inhalte an zwei Referenzszenarien dargestellt. Nach der technischen und wirtschaftlichen Bewertung der entwickelten Konzepte folgen eine Zusammenfassung der Ergebnisse und ein Ausblick zukünftiger Weiterentwicklungen.

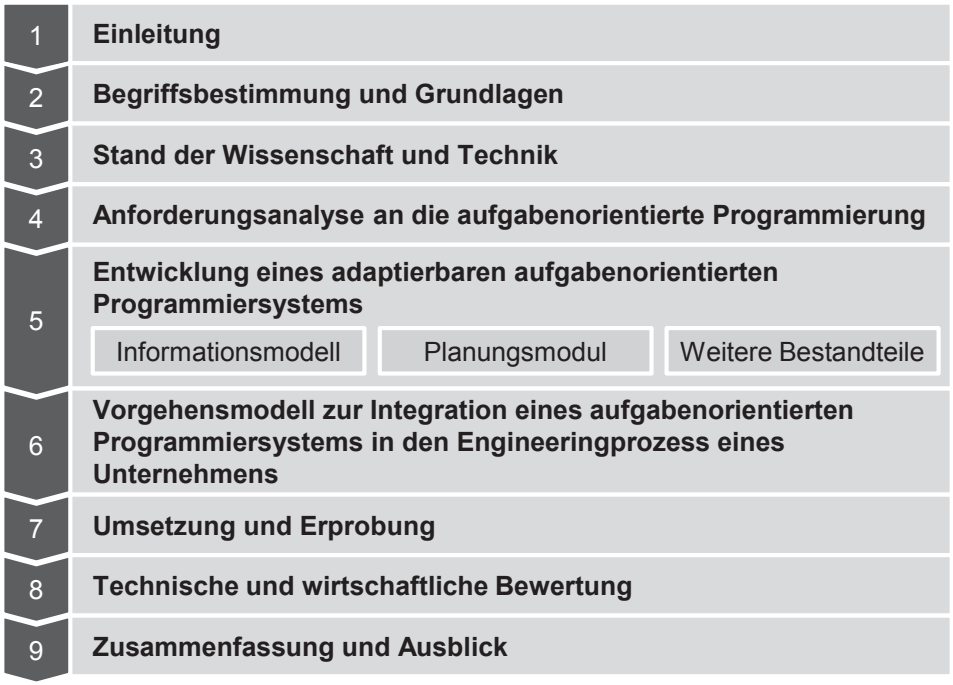


Abbildung 2: Aufbau der Arbeit

2 Begriffsbestimmung und Grundlagen

2.1 Begriffsdefinitionen

System

In dieser Arbeit wird die in der VDI-RICHTLINIE 3633 festgelegte Definition für das Umfeld der Produktionstechnik verwendet. Darin wird ein System in Bezug auf die DIN IEC 60050-351 definiert als „(…) eine von ihrer Umwelt abgegrenzte Menge von Elementen, die miteinander in Beziehung stehen“ (VDI-RICHTLINIE 3633 2014). Ein System ist durch eine Grenze zur Umwelt gekennzeichnet, über die durch Schnittstellen Ein- und Ausgangsgrößen wie Materie, Energie oder Information ausgetauscht werden können. Ein System lässt sich außerdem in Teilsysteme (Subsysteme) und nicht weiter zerlegbare Bestandteile (Systemelemente) aufspalten. Die Elemente im System sind durch Zustandsgrößen und Zustandsübergänge beschrieben.

Modell und Modellierungssprache

Ein Modell wird in der VDI-RICHTLINIE 3633 als „(…) eine vereinfachte Nachbildung eines geplanten oder existierenden Systems mit seinen Prozessen in einem anderen begrifflichen oder gegenständlichen System“ definiert (VDI-RICHTLINIE 3633 2014). Als Modellierungssprache wird ein standardisiertes Medium für die Kommunikation bezeichnet, welches durch definierte Regeln die Elemente und Beziehungen des jeweiligen Modells eindeutig macht (DELLIGATTI 2014, S. 5). Beispiele für Modellierungssprachen sind die Unified Modeling Language (UML) oder Petri Netze.

Informationsmodell

SCHENCK & WILSON (1994) definieren ein Informationsmodell als formale Beschreibung von Typen, Ideen, Daten und Prozessen, welche zusammen ein Modell eines Betrachtungsbereichs der realen Welt formen. Die darzustellenden Informationen und deren Strukturierung sind vom Anwendungsfokus des Informationsmodells abhängig (SELIG 2011).

Simulation

Der Begriff Simulation wird in dieser Arbeit nach der VDI-RICHTLINIE 3633 wie folgt definiert: "Simulation ist die Nachbildung eines dynamischen Prozesses in

einem Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind“ (VDI-RICHTLINIE 3633 2014).

Funktionalität und Funktion

SELIG (2011) definiert Funktionalitäten als „(…) abgeschlossene Methoden zur Bearbeitung von Aufgaben“. Funktionen sind Funktionalitäten, welche über Schnittstellen nach außen bereitgestellt werden (SELIG 2011).

Planen

REFA definiert Planen als „(…) das systematische Suchen und Festlegen von Zielen sowie von Aufgaben und Mitteln zum Erreichen dieser Ziele“ (REFA 1985, S. 18). In Bezug auf Robotersysteme, kann das Planen als die Analyse der notwendigen Aktivitäten für eine Problemstellung und die Bestimmung deren Durchführung konkretisiert werden (HAUN 2007, S. 127).

Steuern

Unter Steuern ist der Vorgang zu verstehen, bei dem durch variable Eingangsgrößen die variablen Ausgangsgrößen eines Systems auf Grund dessen Gesetzmäßigkeiten, gezielt beeinflusst werden. Im Unterschied zur Regelung erfolgt beim Steuern keine Rückführung der regelnden Größe und der Vergleich mit dem Führungswert (DIN IEC 60050-351 2014).

Gerät und Ressource

Im Bereich der Automatisierungstechnik wird unter Gerät ein abgeschlossener Bestandteil eines Automatisierungssystems verstanden (FISCHER 1981). „Es kann sich hierbei entweder um einen Sensor, einen Aktor oder eine Steuerung wie NC (Numerical Control), SPS (Speicherprogrammierbare Steuerung), MC (Motion Control) oder RC (Roboter Control) handeln. Ein Gerät kann Funktionen nach außen bereitstellen und eine interne Datenhaltung haben“ (SELIG 2011, S. 29). In dieser Arbeit wird teilweise synonym für alle möglichen Bestandteile eines Automatisierungssystems, d. h. auch für Werkstückträger oder Schutzzäune, der Begriff Ressource verwendet.

Programm und Programmieren

Nach DIN IEC 60050-351 ist ein Programm „(…) eine nach den Regeln der verwendeten Sprache festgelegte syntaktische Einheit aus Anweisungen und Vereinbarungen, welche die zur Lösung einer Aufgabe notwendigen Elemente umfaßt“ (DIN IEC 60050-351 2014). „Das Programmieren umfasst die Tätigkei-

ten des Entwerfens, Codierens und Testens eines Programms“ (DIN IEC 60050-351 2014).

Programmiersystem

Unter einem Programmiersystem ist im Rahmen dieser Arbeit ein Softwaresystem zur Programmentwicklung zu verstehen (PAHL 1990). Es kann aus weiteren Teilprogrammen wie Benutzerschnittstellen oder Simulationssystemen zusammengesetzt sein.

Befehl

Befehle geben Arbeitsschritte an und stellen bezüglich einer Programmiersprache die kleinste nicht weiter zerlegbare Einheit dar (CLAUS & SCHWILL 2003).

Kommunikationsschnittstellen und Kommunikationsobjekte

In dieser Arbeit werden unter Kommunikationsschnittstellen bzw. Schnittstellen Objekte verstanden, die einen Zugriff auf die Funktionalitäten von Ressourcen ermöglichen. Der Begriff kann daher auch als Synonym zum Begriff Kommunikationsobjekt gebraucht werden, wie er von SELIG (2011) verwendet wird. Dieser definiert ein Kommunikationsobjekt als Objekt innerhalb eines Gerätes, auf das andere Kommunikationsteilnehmer von außen zugreifen können (SELIG 2011, S. 30).

Konfigurieren

In der DIN IEC 60050-351 wird Konfiguration als das Zusammensetzen von Funktions- oder Baueinheiten zu einem Regelungs- und Steuerungssystem verstanden. Im Rahmen diese Arbeit wird der Begriff Konfiguration auf die Kommunikationstechnik in Automatisierungssystemen bezogen. Dies umfasst die physikalische Signalverbindung, die Konfiguration der Datenschnittstellen zwischen beteiligten Systemkomponenten und die Erstellung gerätespezifischer Funktionen (KRUG 2013, S. 2).

Montage und Montageplanung

Die Montage ist der letzte Schritt in der industriellen Produktion vor der Auslieferung des Produkts an den Kunden (LOTTER & WIENDAHL 2012, S. 1). In vorhergehenden Produktionsschritten wurden mit unterschiedlichsten Verfahren die Produkteinzelteile gefertigt. Die Aufgabe der Montage ist es, die einzelnen Bauteile zu einem funktionsfähigen Produkt zusammenzusetzen (LOTTER & WIENDAHL 2012, S. 1). Die Montageplanung wird in Anlehnung an GRUNWALD

(2002, S. 11) als ein Vorgehen definiert, welches die Aufgabe hat, Montageanlagen und Montageabläufe zu entwickeln.

Skill bzw. Fähigkeit

In der Literatur werden unter dem Begriff Fähigkeit bzw. Skill unterschiedliche Begrifflichkeiten verstanden (MORROW & KHOSLA 1997; BENGEL 2008; KLUGE 2011; SMALE 2011), weswegen zunächst eine umfassende Definition herausgearbeitet wird. Als Ausgangsbasis dient die Definition nach CÂNDIDO & BARATA (2007). Diese verstehen Skills als das Vermögen eine Aktivität oder Operation auszuführen, die den Produktionsprozess unterstützt. Diese Definition wird weiter detailliert: Skills sind eine semantische und herstellerunabhängige Beschreibung der über Befehle, Funktionen, Funktionsbausteine oder Schnittstellen bereitgestellten Funktionalitäten einer Ressource, welche den Montageprozess unterstützt. Skills sind somit Ressourcen zugehörig und sind aus Sicht einer abstrahierten Ressource beschrieben.

Reifegradmodell

„Reifegradmodelle verfolgen das Ziel, die Arbeitsabläufe eines Unternehmens zu analysieren und zu optimieren. Hierzu werden die Prozesse einer Organisation anhand einer Systematik auf Projekt- und Unternehmensebene bewertet und die gewonnen Ergebnisse anschließend in Maßnahmen zur Prozessverbesserung umgesetzt“ (HOFFMANN 2013, S. 492).

Vorgehensmodell

„Ein Vorgehensmodell unterteilt ein Entwicklungsvorhaben in einzelne Aktivitäten einschließlich ihrer logischen Abfolge und regelt die Verantwortlichkeiten der beteiligten Personen (Rollen). Es gibt eine Antwort auf die Frage: ‚Wer macht was wann’ in einem Entwicklungsprojekt“ (BENDER ET AL. 2005, S. 384).

Entwicklungs- und Engineeringprozess

Als Entwicklungsprozesse werden im Rahmen dieser Arbeit die Prozesse bei der Entwicklung eines Produktes oder einer Produktionsanlage verstanden. Dabei werden Rollen, Abläufe und Vorgehensweisen festgelegt (SCHEDL 2009, S. 14). In Bezug auf Produktionsanlagen wird auch häufig der Begriff Engineering Prozess verwendet. Im Vergleich mit der Definition der Montageplanung ist erkennbar, dass sich die beiden Begriffe überschneiden. Im Rahmen dieser Arbeit wird daher, wenn der gesamte Prozess bei der Entwicklung einer Montageanlage betrachtet wird, vom Engineeringprozess gesprochen. Sind die speziellen Tätigkei-

ten bei der Planung eines Montagesystems, wie z. B. die Ableitung einer Montagereihenfolge, adressiert, wird der Begriff Montageplanung verwendet.

Geschäftsprozess

„Ein Geschäftsprozess besteht aus der funktions- und organisationsübergreifenden Verknüpfung wertschöpfender Aktivitäten, die von Kunden erwartete Leistungen erzeugen und die aus der Geschäftsstrategie abgeleiteten Prozessziele umsetzen“ (SCHMELZER & SESSELMANN 2008, S. 64).

2.2 Entwicklungsprozesse automatisierter Anlagen

2.2.1 Vorgehensmodelle und Methoden für die Entwicklung technischer Systeme

In den vergangenen Jahrzehnten wurden verschiedene Ansätze zur Unterstützung der Entwicklung von technischen Systemen entwickelt. Im Gegensatz zu Methoden der Montageplanung, die in Abschnitt 2.6 vorgestellt werden, sind diese allgemein für technische Systeme anwendbar. Erste Vorgehensmodelle aus der Softwareentwicklung, wie das Wasserfallmodell, waren durch einen seriellen Ablauf geprägt, d. h. Ergebnisse werden erst bei einem gewissen Reifegrad an die nächste Abteilung weitergegeben (DRÖSCHEL ET AL. 1997).

Ein allgemeines Vorgehen beim Entwickeln und Konstruieren beschreibt beispielsweise die VDI-RICHTLINIE 2221. Das Vorgehen ist in insgesamt sieben Aufgaben unterteilt, die dem Entwickler helfen sollen, die Komplexität des Entwicklungsprojektes überschaubar zu halten. Sie fokussiert jedoch den Bereich der Mechanik. Da das Vorgehen mit dem erhöhten Einsatz von Elektronik und Software an seine Grenzen stößt, wurde die VDI-RICHTLINIE 2206 zur Entwicklung mechatronischer Systeme definiert. Mechatronik ist als interdisziplinäres Gebiet zu verstehen, bei dem mechanische, elektronische und informationstechnische Systeme zusammenwirken und die Software die Funktion bestimmt (ISERMANN 2008, S. 18), wobei jedoch weitere Definitionen existieren. Weitere Vorgehensmodelle und Methoden wie das W-Modell oder agile Methoden wie ‘Extrem Programming‘ und SCRUM werden z. B. von HENSEL (2011) vorgestellt. Dieser entwickelt auf Basis des V-Modells das Vi-Modell mit dem Ziel, einen modellbasierten, simulationsgestützten Entwicklungsprozess von Automatisierungsprojekten zu ermöglichen. Eine sehr umfangreiche Vorstellung über-

greifender und disziplin-spezifischer Vorgehensmodelle, erfolgt in EIGNER ET AL. (2014, S. 15–52).

2.2.2 Digitale Werkzeuge in Entwicklungsprozessen

Ein Überbegriff für den Einsatz digitaler Werkzeuge in Entwicklungsprozessen ist das Digitale Engineering. Dieses ist definiert als „(…) die durchgängige Nutzung digitaler Methoden und Werkzeuge über den Produktentstehungs- und Produktionsprozess (…)“ (SCHUMANN ET AL. 2011). Der Einsatz dieser Ansätze kann auch unter dem Oberbegriff digitale Fabrik zusammengefasst werden, welche nach VDI-RICHTLINIE 4499 wie folgt definiert ist:

„Die digitale Fabrik ist der Oberbegriff für ein umfassendes Netzwerk von digitalen Modellen, Methoden und Werkzeugen – u. a. der Simulation und dreidimensionalen Visualisierung – die durch ein durchgängiges Datenmanagement integriert werden. Ihr Ziel ist die ganzheitliche Planung, Evaluierung und laufende Verbesserung aller wesentlichen Strukturen, Prozesse und Ressourcen der realen Fabrik in Verbindung mit dem Produkt.“ (VDI-RICHTLINIE 4499 2008, S. 3).

In den Phasen des Entwicklungsprozesses kommen mannigfaltige Werkzeuge für die verschiedenen Anwendungsgebiete zum Einsatz. Im Folgenden soll ein kurzer Überblick der wichtigsten Vertreter und deren Zusammenwirken beschrieben werden. Die eingesetzten Werkzeuge lassen sich im Wesentlichen in die Folgenden Klassen unterteilen (BRACHT ET AL. 2009):

Als *Erzeuger und Autorensysteme* werden CAx- und Office-Werkzeuge eingesetzt. Typische CAx-Werkzeuge sind nach BRACHT ET AL. (2009):

- CAD (Computer Aided Design)-Werkzeuge zur „(…) Erzeugung der geometrischen, technologischen und funktionalen Daten eines Produktes oder Produktionsmittels in der Entwicklungsphase“ (BRACHT ET AL. 2009, S. 180),
- CAE (Computer Aided Engineering)-Werkzeuge „(…) zur computerunterstützten Berechnung, Simulation und Analyse“ (BRACHT ET AL. 2009, S. 182),
- CAP (Computer Aided Planning)-Werkzeuge zur rechnergestützte Arbeits-planung) bzw. CAPP (Computer Aided Process Planning)-Werkzeuge und
- CAM (Computer Aided Manufacturing)-Werkzeuge zum computergestützten Montieren und Fertigen.

Als *Integrationsplattform bzw. Datenbackbone* der Erzeuger und Autorensysteme werden PDM/EDM-Systeme (PDM – Product Data Management, EDM – Engineering Data Management) eingesetzt (BRACHT ET AL. 2009). Die Aufgabe dieser Systeme ist es, „(…) Informationen über Produkte und deren Entstehungsprozesse bzw. Lebenszyklen konsistent zu speichern, zu verwalten und transparent für alle relevanten Bereiche eines Unternehmens bereitzustellen“ (VDI-RICHTLINIE 2219 2002, S. 4). Werkzeuge zur *Digital Engineering Visualisation* (DEV) werden zur Präsentation und Diskussion von Analyseergebnissen und Berechnungen verwendet (BRACHT ET AL. 2009). Die Einordnung der digitalen Werkzeuge in eine modellbasierte virtuelle Produktentwicklung wird z. B. auch von EIGNER ET AL. (2014) beschrieben.

2.3 Einordnung und Bestandteile der Montage

Nach SPUR & KRAUSE (1997) lässt sich die Montage in den Kreislauf von Produktentstehung und Produktmarkt eingliedern. Ausgehend von der Produktidee erfolgt die Konstruktion, woran sich die Arbeitsvorbereitung anschließt. Nach der darauf folgenden Fertigung, werden die erzeugten Einzelteile in der Montage zum fertigen Produkt zusammengesetzt. Über den Vertrieb kommt dieses Produkt in den Markt, worauf hin wieder neue Produktideen oder -änderungen entstehen können (SPUR & KRAUSE 1997). Wie in den Definitionen beschrieben, stellt die Montage folglich den letzten Schritt der Produktionsentstehung dar, bevor die fertigen Produkte über den Vertrieb an den Kunden gehen (LOTTER & WIENDAHL 2012).

Die wesentlichen Vorgänge der Montage sind das Fügen sowie die Funktionen zur Handhabung von Werkstücken, die nach DIN 8593 bzw. VDI-RICHTLINIE 2860 definiert sind. Die Tätigkeiten des Justierens, des Kontrollierens sowie der Sonderoperationen, wie Erwärmen oder Reinigen, werden auch der Montage zugeordnet (LOTTER & WIENDAHL 2012). Als Fügen wird „das auf Dauer angelegte Verbinden oder sonstige Zusammenbringen von zwei oder mehr Werkstücken geometrisch bestimmter Form oder von ebensolchen Werkstücken mit formlosem Stoff“ (DIN 8593 2003, S. 3) definiert. „Dabei wird jeweils der Zusammenhalt örtlich geschaffen und im Ganzen vermehrt“ (DIN 8593 2003, S. 3). Die Vorgänge der Montage werden zudem oftmals als Montageprozesse bezeichnet. Abbildung 3 zeigt eine Übersicht aller Funktionen der Montage.

An dieser Stelle sei darauf hingewiesen, dass in der DIN 8593 Fertigungsverfahren unterschieden werden, die nicht die gleiche Beschreibungsebene mit den in

der VDI-RICHTLINIE 2860 definierten Funktionen der Handhabung haben. So beschreibt die DIN 8593 Zustandsänderungen von mindestens zwei Bauteilen, deren Zusammenhalt durch unterschiedliche Arten, z. B. Kraft- oder Formschluss bzw. Energieformen, erzeugt werden und Bewegungen enthalten können. Hingegen sind in der VDI-RICHTLINIE 2860 elementare und zusammengesetzte, lösungsneutrale Funktionen beschrieben, die sich ausschließlich auf die Bewegung, das Halten und Lösen, sowie das Prüfen oder Messen von Eigenschaften einzelner oder von Gruppen von Körpern beziehen. In der VDI-RICHTLINIE 2860 sind darüber hinaus Symbole für die einzelnen Funktionen definiert und es werde sieben Elementar funktionen unterschieden (Teilen, Vereinigen, Drehen, Verschieben, Halten, Lösen, Prüfen). Bezugsgrößen wie Position, Orientierung oder Ordnungszustand, welche in der VDI-RICHTLINIE 2860 beschrieben sind, werden für die Fertigungsverfahren in der DIN 8593 nicht genannt.

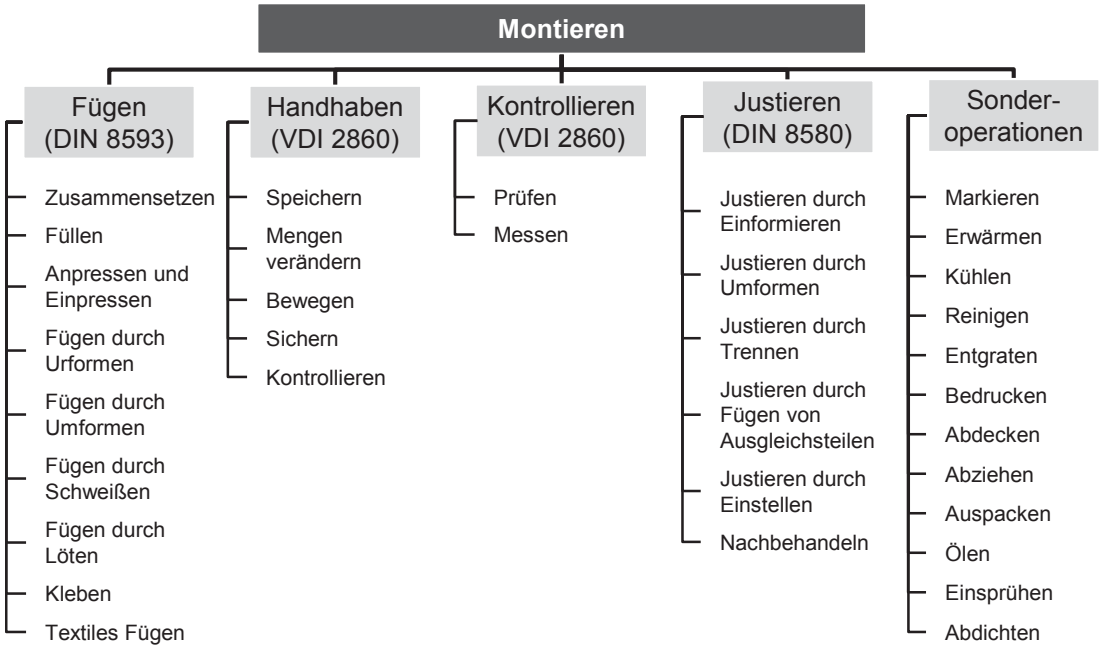


Abbildung 3: Funktionen der Montage (LOTTER & WIENDAHL 2012)

Montagevorgänge lassen sich darüber hinaus in Primärprozesse bzw. Primärvorgänge und Sekundärprozesse bzw. Sekundärvorgänge einteilen. Als Primärvorgänge sind alle Aufwendungen definiert, die zum Vervollständigen eines Produktes und damit zur Wertschöpfung beitragen (LOTTER & WIENDAHL 2012, S. 49). Ein Beispiel ist das Zusammensetzen von Bauteilen zur Vervollständigung des Produktes. Sekundärvorgänge sind alle unterstützenden Aufwendungen, die notwendig sind, aber keine Wertschöpfung des Produktes bewirken. Beispiele

hierfür sind das Weitertransportieren oder das Umgreifen während eines Handhabungsvorgangs (LOTTER & WIENDAHL 2012, S. 49).

2.4 Automatisierte Montagesysteme

Systeme, die zwei oder mehrere Bauteile verbinden und somit Montageoperationen verrichten, werden als Montagesysteme definiert (SCHMIDT 1992, S. 10). In Abhängigkeit von den Anforderungen der Montageaufgabe nach Investition, Stückzahlen und Flexibilität kommen unterschiedliche Montagesysteme, beginnend mit der manuellen Lösung, über die halbautomatische (hybride) Lösung bis hin zur vollautomatischen Lösung, sowie Übergangsformen zum Einsatz (LOTTER & WIENDAHL 2012). Im Rahmen dieser Arbeit werden die vollautomatischen Montagesysteme fokussiert und im Folgenden näher betrachtet.

2.4.1 Bestandteile

Die Bestandteile eines Montagesystems sind nach SCHMIDT (1992, S. 20) die Elemente, welche innerhalb von Montagesystemen Funktionen oder Prozesse zur Erfüllung der Montageaufgabe ausführen. Im Umfeld der Montage existieren gleichrangig verwendete Begriffe, wie Betriebsmittel und Produktionsmittel (LOFERER 2002, S. 14). In dieser Arbeit wird durchgängig der Begriff Ressource verwendet. SCHMIDT (1992, S. 21) unterscheidet die Bestandteile automatisierter Montagesysteme anhand ihrer Komplexität in drei Kategorien:

- Primitive Prozessträger wie Linearachsen oder Druckzylinder,
- einzelne Montagekomponenten, wie einzelne Sensoren oder Spannvorrichtungen und
- kombinierte Montagekomponenten vereinen mehrere Funktionen und damit Komponenten in sich, z. B. Industrieroboter.

Für die Strukturierung von Ressourcen existieren verschiedene Konzepte, wie z. B. die VDI-Richtlinie 2860 für Handhabungseinrichtungen. In dieser Arbeit wird eine Einordnung basierend auf den Einteilungen nach SECKNER (2008) und LOFERER (2002) verwendet. Abbildung 4 zeigt die daraus abgeleitet Einordnung mit mehreren Beispielen.

Handhabung- bzw. Bewegungseinrichtungen sind nach der VDI-RICHTLINIE 2860 als technische Einrichtungen definiert, die Bewegungen durchführen und dabei Werkstücke oder Werkzeuge bewegen. Beispiele sind Industrieroboter, Einlegegeräte, oder Dreheinrichtungen. Besondere Relevanz haben hierbei In-

dustrieroboter (IR) (WÖRN 2003, S. 53). Diese sind definiert als „(…) universell einsetzbare Bewegungsautomaten mit mehreren Achsen, deren Bewegungen hinsichtlich Bewegungsfolgen und Wegen bzw. Winkeln frei (d. h. ohne mechanischen Eingriff) programmierbar und gegebenenfalls sensorgeführt sind“ (VDI-RICHTLINIE 2860 1990).

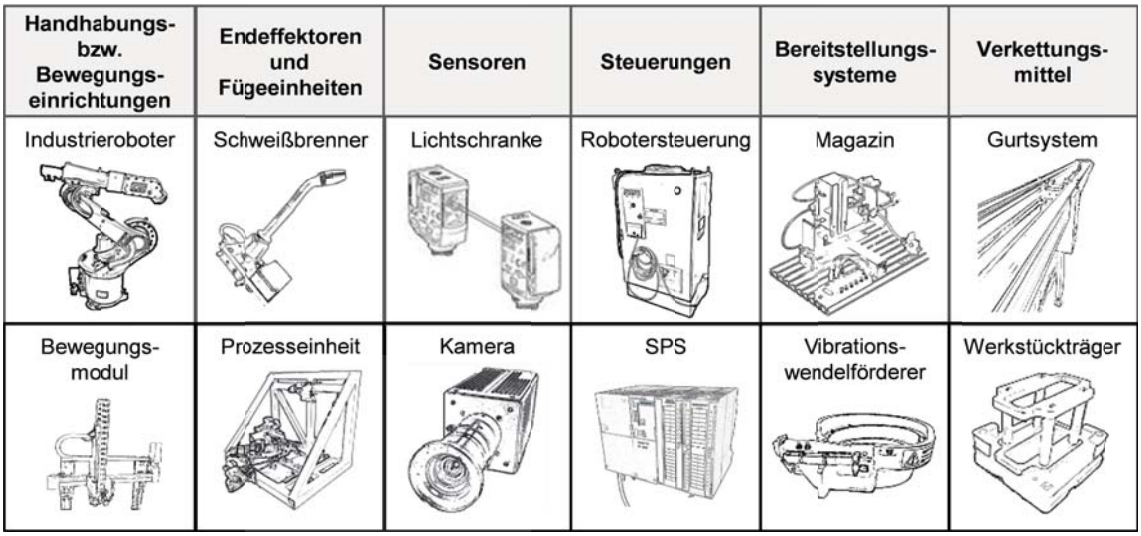


Abbildung 4: Ausgewählte Bestandteile automatisierter Montagesysteme in Anlehnung an LOFERER (2002)

Wegen der besonderen Bedeutung der Industrieroboter für Montagesysteme (WÖRN 2003, S. 53), werden deren Aufbau und Eigenschaften im Folgenden genauer beschrieben. Abbildung 5 zeigt den typischen Aufbau eines Knickarmroboters mit sechs Freiheitsgraden. Weitere gängige Roboterarten sind SCARA oder Portalroboter (LOTTER & WIENDAHL 2012). Für die Bewegungen des Roboters hat der Arbeitspunkt des Werkzeugs besondere Relevanz, der auch als Tool-Center-Point (TCP) bezeichnet wird. Der TCP „(…) stellt den Werkzeugarbeitspunkt dar, d. h. den Wirkort als Bezugspunkt der Schnittstelle zwischen Werkzeug und Werkstück“ (GERKE 2014, S. 153). Das im TCP befindliche Koordinatensystem wird nach DIN EN ISO 8373 als TCP-Koordinatensystem bezeichnet. Dieses verschiebt sich abhängig vom eingesetzten Werkzeug. Der durch die mit dem TCP erreichbaren Positionen aufgespannte Raum, wird als Arbeitsraum bezeichnet. Die Kombination von Position und Orientierung im Raum wird laut DIN EN ISO 8373 als Pose bezeichnet. Synonym kann auch der Begriff Frame verwendet werden. Orientierungen eines Frames werden meist in Euler Winkeln beschrieben (WEBER 2009). Um die Bewegungen des TCP eines IR zu einem bestimmt Punkt im Arbeitsraum zu beschreiben, werden Koordinatensysteme in den Gelenken definiert. Damit lassen sich Transformationsmatrizen dieser Koor-

dinatensysteme definieren, welche Bezugssysteme für die Vorwärts- und Rückwärtstransformation darstellen (REINISCH 1992, S. 34). Rechenvorschriften wie nach Denavit-Hartenberg vereinheitlichen die Berechnung und ermöglichen es, die kinematische Modellierung zu automatisieren (REINISCH 1992, S. 35).

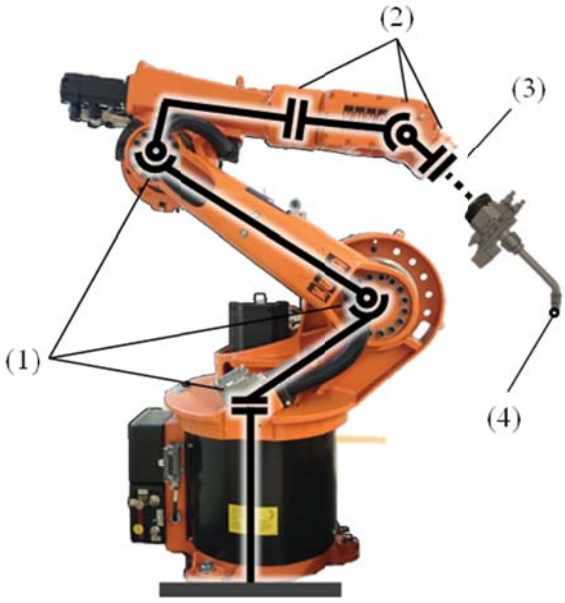


Abbildung 5: Typischer Aufbau eines Sechsachs-Industrieroboters: (1) Hauptachsen, (2) Handachsen, (3) Flansch für Werkzeugaufnahme, (4) Werkzeug mit TCP an der Spitze

In Rahmen dieser Arbeit werden *Endeffektoren* und *Fügeeinheiten* zu einer Kategorie zusammengefasst. Charakteristisch für *Endeffektoren* ist die Verbindung zu Handhabungseinrichtungen, durch die sie geführt werden. Im Zusammenspiel mit diesen ermöglichen sie Zustandsänderungen zum Fügen oder Bewegen von Bauteilen. Endeffektoren werden in Greifer und Werkzeuge (z. B. Schweißzange oder Lackierpistole) unterschieden (LOFERER 2002). *Fügeeinheiten* sind im Gegensatz zu Endeffektoren kombinierte Komponenten, die Zustandsänderungen zum Fügen von Bauteilen und Baugruppen durchführen.

Sensoren sind in Montagesysteme integriert, um z. B. Qualitätsprüfungen oder Anwesenheitskontrollen durchzuführen. Für die Detektion sich bewegender Elemente kommen zumeist optische Sensoren zum Einsatz (SECKNER 2008, S. 17). Intelligente Sensoren oder *Sensorsysteme* verfügen darüber hinaus auch über eine Einheit zur Weiterverarbeitung der Signale (WENK 2002).

Steuerungen bzw. Steuerungseinrichtungen koordinieren die Komponenten des Montagesystems, um den programmierten Montageprozess abzubilden (LOFERER

2002). Sie sind physisch nicht am Montageprozess beteiligt (SECKNER 2008, S. 50) und wie folgt aufgebaut. Bedienungseingaben bzw. Signale und Rückgabewerte der Anlage werden in einer internen Signalverarbeitung nach vordefinierten Regeln und Vorgaben verarbeitetet und daraus Ausgangssignale an die Anlage abgeleitet (VDI/VDE TECHNISCHE REGEL 3683). Beispiele sind Robotersteuerungen, Speicherprogrammierbare Steuerungen, embedded Controller oder NC-Steuerungen.

Mit *Bereitstellungssystemen* werden einzelne Bauteile oder Baugruppen für den Montageprozess zur Verfügung gestellt. Beispiele sind Vibrationswendelförderer oder Magazine (LOFERER 2002).

Verkettungsmittel sind zusammen mit den *Flusssteuerelementen* für den Materialfluss in Montageanlagen verantwortlich (SECKNER 2008; LOFERER 2002). Typische Arten sind Kettenförderer oder Doppelgurtförderer (LOFERER 2002) bzw. Stopper und Weichen (SECKNER 2008).

2.4.2 Steuerungsstrukturen und industrielle Kommunikationstechnik

Im Folgenden wird eine Einteilung der Steuerungsstrukturen vorgestellt, welche nicht nur für Montagesysteme gültig ist, sondern allgemein die Hierarchieebenen der Rechner- und Steuerungssysteme von Produktionsanlagen beschreibt. Die Strukturierung orientiert sich an den in der DIN EN 62264-1 definierten Ebenen und wird als Automatisierungspyramide bezeichnet (VOGEL-HEUSER & WANNAGAT 2009). Abbildung 6 zeigt die typischen Ebenen der Automatisierungspyramide.

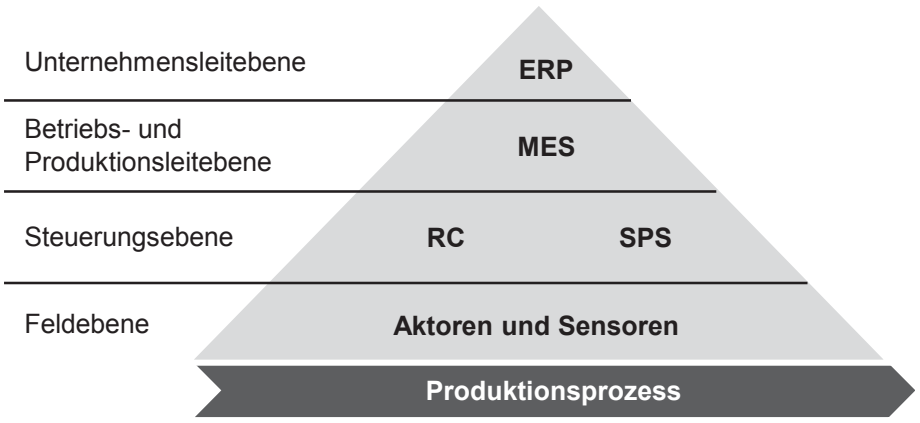


Abbildung 6: Ebenen der Automatisierungspyramide in Anlehnung an DIN EN 62264 und HEINRICH ET AL. (2015)

Aktuelle Beiträge sehen in der Zukunft eine Auflösung der Ebenen durch die 4. Industrielle Revolution (KRÜCKHANS & MEIER 2013), (BAUERNHANSL 2014), deren Realisierbarkeit ist jedoch noch zu prüfen (FORSTNER & DÜMMLER 2014).

Um die Kommunikation der verschiedenen Komponenten in einem Montagesystem zu ermöglichen, kommen im industriellen Umfeld Kommunikations- und Informationsnetzwerke zum Einsatz. Für deren Einordnung kann das OSI Referenzmodell (ISO/IEC 7498) verwendet werden. Um für den industriellen Einsatz verwendbar zu sein, müssen diese zeitliche Anforderungen einhalten, die von der Einordnung in der Automatisierungspyramide abhängen und nach unten hin niedrigere Antwortverzögerungen erfordern (KRUG 2013, S. 18). Typische Beispiele für industriell verwendete Feldbuse auf Steuerungs- und Feldebene sind EtherCAT, SERCOS oder PROFIBUS. KRUG (2013), SCHNELL (2013) und SELIG (2011) stellen einen umfassenden Überblick der industriellen Kommunikation mit den verwendeten Protokollen, Topologien und Spezifikationen sowie den vorhandenen Rahmenbedingungen vor. Im Rahmen dieser Arbeit wird daher auf eine detailliert Beschreibung verzichtet.

2.5 Programmierung von automatisierten Montagesystemen

Vor der Inbetriebnahme und dem Betrieb müssen automatisierte Montagesysteme programmiert werden, um in den Programmen die automatisch auszuführenden Vorgänge des Montageprozesses zu beschreiben (FANDEL ET AL. 1994). Die Methoden zur Programmierung werden im Folgenden beschrieben, unterteilt in die programmierbaren Gerätetypen SPS und Industrieroboter, welche typischerweise in Montagesystemen vorkommen (SECKNER 2008, S. 20).

2.5.1 Speicherprogrammierbare Steuerungen (SPS)

Als Basis für die Programmierung von Speicherprogrammierbaren Steuerungen (SPS) wurde von der International Electronical Comission (IEC) der Standard DIN EN 61131 erarbeitet, der in Teil 3 fünf Programmiersprachen definiert. Die Norm sieht sich als Richtlinie und nicht als starre Vorgabe, was dazu führt, dass SPS-Hersteller nur einen unterschiedlich großen Teil der Norm umsetzten (JOHN & TIEGELKAMP 2009). Die fünf Programmiersprachen können in drei textuelle Sprachen (*Anweisungsliste (AWL)*, *Strukturierter Text (ST)* und der *Ablaufsprache (AS)* in der textuellen Variante) sowie drei graphische Sprachen (*Kontaktplan (KOP), Funktionsbausteinsprache (FBS)* und der *Ablaufsprache (AS)* in der graphischen Variante) unterteilt werden (JOHN & TIEGELKAMP 2009, S. 103). Die

DIN EN 61131 definiert zudem Standardfunktionen und Funktionsbausteine (JOHN & TIEGELKAMP 2009, S. 213). Die Anwenderprogramme werden vom Betriebssystem der SPS zyklisch ausgeführt, wobei die Zykluszeit konstant ist. In den drei wesentlichen Schritten wird in einem Zyklus jeweils der Eingangsspeicher aktualisiert, das Anwenderprogramm abgearbeitet und der Ausgangsspeicher geschrieben (LEPERS 2005, S. 22). Für verteilte Steuerungen, wie z. B. agentenbasierte Ansätze, wird in der DIN EN 61499 eine Weiterentwicklung der DIN EN 61131 definiert. Ein wesentlicher Unterschied der DIN EN 61499 ist eine ereignisorientierte Ausführung im Gegensatz zur zyklischen Ausführung in der DIN EN 61131 (SCHWARZ 2009).

SPSen werden zumeist offline an PC-basierten Systemen programmiert und die Programme an die SPS übertragen (SECKNER 2008). In neueren Konzepten wird außerdem der Ansatz verfolgt, Steuerungsprogramme für IEC 61131-3 konforme Steuerungen direkt mit einem integrativen Modell aus dem Entwicklungsprozess zu generieren (WITSCH & VOGEL-HEUSER 2004). Zugehörige Ansätze werden im Stand der Wissenschaft und Technik diskutiert.

2.5.2 Programmierung von Industrierobotern

Für die Programmierung von Industrierobotern existiert eine Vielzahl von verschiedenen Ansätzen. Allgemein lassen sich diese jedoch nach dem Ort der Programmierung in die Kategorien Online- oder Offline-Programmierung (HAUN 2007) sowie nach der Beschreibungsart der Aufgabe in implizite und explizite Verfahren (FELDMANN ET AL. 2014, S. 328) einteilen. Für die Online-Programmierung muss der reale Roboter angeschaltet und verfügbar sein, bei der Offline-Programmierung ist der reale Roboter im Gegensatz dazu während des Programmiervorgangs nicht notwendig. Daraus ergibt sich auch der wesentliche Nachteil der Online-Programmierung, da bei dieser der Roboter nicht für den Produktionsprozess zur Verfügung steht (HAUN 2007). Jedoch ist bei der Offline-Programmierung im Normalfall ein Abgleich mit der Realität notwendig (KUGELMANN 1999, S. 2). Bei einer expliziten Programmierung werden alle notwendigen Informationen des Programmablaufs durch den Programmierer vorgegeben (FELDMANN ET AL. 2014, S. 328). Im Gegensatz dazu muss bei einer impliziten Programmierung ein Programmiersystem eigenständig für eine bestimmte Aufgabe einen später umsetzbaren Lösungsplan erstellen (FELDMANN ET AL. 2014, S. 328). Die Beschreibung der Fertigungsaufgabe erfolgt daher auf einem höheren Abstraktionslevel, eine einheitliche Definition auf welchem Level die Aufgabe beschrieben wird, existiert jedoch nicht (KLEINEIDAM 1990).

WECK & BRECHER (2006) unterteilen die *Online-Programmierung* in Teach-In, Playback, steuerungsgestützte und werkstattorientierte Verfahren. „Unter dem „Teach-In"–Verfahren versteht man alle Verfahren, bei denen einem Roboter sein späteres Verhalten durch „Learning by doing" so lange angelernt wird, bis das gewünschte Ergebnis erzielt ist" (HAUN 2007, S. 424). Beim Teach-In Verfahren wird der Roboter in die gewünschten Positionen verfahren und die Achsstellungen abgespeichert. Im Programm werden diese in gewünschter Reihenfolge abgefahren (SECKNER 2008). Im Gegensatz dazu wird beim Playback Verfahren das Robotersystem direkt geführt und die Bewegungen z. B. in bestimmten Zeitintervallen aufgezeichnet, um dessen Verhalten vorzugeben (WECK & BRECHER 2006). Das Konzept der werkstattorientierten Programmierung lehnt sich an diese Art der Programmierung aus dem Bereich der NC-Maschinen an (WECK & BRECHER 2006). Ein anderer verwendeter Begriff für dieses Verfahren ist „Makroprogrammierung" (HUMBURGER 1998, S. 11). Der Anwender, wie z. B. das Werkstattpersonal, verwendet vorgefertigte Funktionsbausteine, die er parametriert und damit den Programmablauf beschreibt. Damit ist eine Programmierung ohne umfassendes Hintergrundwissen bzgl. der Programmierung des Roboters möglich. Die verwendeten Programmiersysteme sind auf spezifische Anwendungsfälle, wie beispielweise Schweißprozesse, zugeschnitten (WECK & BRECHER 2006).

Nach WECK & BRECHER (2006) lassen sich die *Offline-Programmierverfahren* in textbasierte, programmablaufbasierte, simulationsbasierte und aufgabenorientierte Verfahren einteilen. Die textuelle Programmierung erfolgt zumeist mit steuerungsspezifischen Programmiersprachen in rechnergestützten Programmiersystemen. Bei der programmablaufbasierten Programmierung erfolgt, ähnlich wie bei der werkstattorientierten Programmierung, die Beschreibung der Steuerungsaufgabe mit Ablaufdiagrammen und den darin enthaltenen parametrisierbaren Bausteinen für die Roboterbewegungen (WECK & BRECHER 2006). Besonders für die Programmierung komplexer Bearbeitungsvorgänge eignen sich simulationsbasierte bzw. CAD-basierte Programmierwerkzeuge. Um diese Art der Programmierung zu ermöglichen, ist die Nachbildung der Roboterzelle inklusive Kinematik, Dynamik, Bewegungsbereiche der Achsen sowie der Werkstücke notwendig. Zusätzlich ist auch eine Modellierung der Steuerung notwendig, da sich die Steuerungen unterschiedlicher Hersteller bzgl. ihres Verhaltens, z. B. bei Singularitäten, unterscheiden (WECK & BRECHER 2006).

Unter dem Begriff aufgabenorientierte Programmierung kann im Allgemeinen eine implizite Programmierung verstanden werden (SECKNER 2008, S. 11). Bei

dieser Art der Programmierung wird nicht mehr beschrieben, „wie“ eine Fertigungsaufgabe abgearbeitet werden soll, sondern „was“ die Fertigungsaufgabe ist (FREUND ET AL. 1990). Damit verfolgt die aufgabenorientierte Programmierung auch das Ziel der Automatisierung der Programmierung (HUMBURGER 1998, S. 12). Aufgabenorientierte Programmiersysteme sind dadurch charakterisiert, dass diese ausgehend von Beschreibungen des Ist- und Sollzustandes und einer Abbildung der Roboterzelle die erforderlichen Bewegungen und Steuerbefehle automatisch generieren (WECK & BRECHER 2006). Die Durchführung von einzelnen Teilaufgaben im Planungsprozess wird durch mehrere Planungsmodule übernommen.

Bei der Gegenüberstellung unterschiedlicher Programmierverfahren ist ersichtlich, dass die aufgabenorientierte Programmierung, insbesondere bei nicht verfügbaren Spezialisten sowie regelmäßigen Produktionsumstellungen oder einer hohen Variantenvielfalt, Vorteile im Vergleich zu anderen Verfahren hat (GOTTSCHALD 2001). Wegen der besonderen Relevanz für diese Arbeit wird die aufgabenorientierte Programmierung im folgenden Abschnitt genauer betrachtet und deren grundlegenden Konzepte erläutert.

2.5.3 Grundlagen der aufgabenorientierten Programmierung

Die wesentlichen Bestandteile eines aufgabenorientierten Programmiersystems sind das Aufgabenmodell, das Umweltmodell, sowie das Planungsmodul (KLEINEIDAM 1990, S. 44). In Abbildung 7 ist die schematische Struktur bei der aufgabenorientierten Roboterprogrammierung für Montageaufgaben dargestellt.

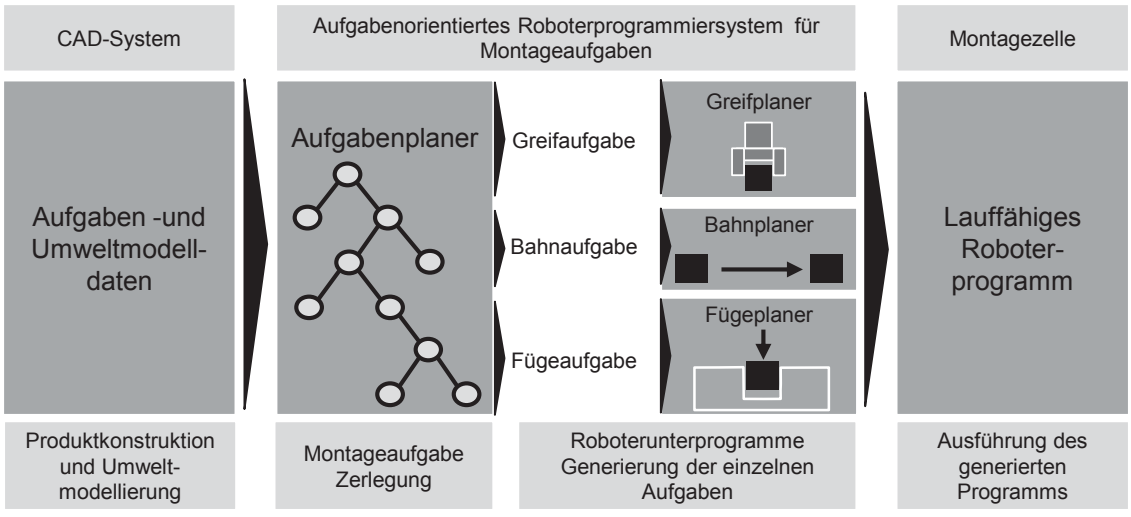


Abbildung 7: Struktur der aufgabenorientierten Programmierung in Anlehnung an WECK & BRECHER 2006

Dieser grundsätzliche Aufbau ist in einer Vielzahl von Konzepten zu finden (KLEINEIDAM 1990; HUMBURGER 1998; SECKNER 2008; MUNZERT 2010; HATWIG 2014). Das Welt- oder Umweltmodell beinhaltet alle Informationen bzgl. der Betriebs-mittel wie Roboter, Effektoren oder Sensoren und der Umgebung der Produktionszelle (HUMBURGER 1998; WEBER 2009). Für die Repräsentation im Umweltmodell können verschiedene Möglichkeiten, wie relationale Modelle oder Netzwerkmodelle, verwendet werden (DILLMANN & HUCK 1991, S. 190). Eine besondere Rolle spielt die Abbildung der Funktionalitäten der Ressourcen der Produktionszellen (SECKNER 2008, S. 61). Im Aufgabenmodell sind Zielzustände und Aktionsfolgen beschrieben (HUMBURGER 1998; WEBER 2009), diese können z. B. direkt aus den CAD-Daten der Werkstücke abgeleitet werden (REINHART ET AL. 2008). Das Aufgaben- und Umweltmodell wird vom Planungsmodul bzw. vom Aufgabenplaner verwendet, um die vorgegebene Aufgabe zunächst in einzelne Aufgaben zu zerlegen. Greif-, Bahn- und Fügeplaner bearbeiten z. B. die einzelnen Montageaufgaben (SECKNER 2008, S. 27).

Kernaufgabe der Greifplanung ist die Identifikation von Greifpositionen am Bauteil unter Berücksichtigung von Größen wie Zugänglichkeit oder Werkstoffeigenschaften des Bauteils und des Greifers. Die Bahnplanung ist zuständig für die automatische Generierung kollisions- und singularitätsfreier Roboterbewegungen. Der Fügeplaner übernimmt die detaillierte Konzeption der Fügeoperationen (WECK & BRECHER 2006). Konzepte zur Beherrschung von Unsicherheiten wie Lagetoleranzen sind wichtig für den Einsatz der aufgabenorientierten Programmierung in der Produktion. Die Unsicherheiten können z. B. durch Sensorik aufgelöst werden. KUGELMANN (1999) und WENK (2002) beschreiben Ansätze zur Beherrschung der Unsicherheiten bei der aufgabenorientierten Programmierung. DILLMANN & HUCK (1991) definieren eine Aufgabenzerlegungshierarchie bei der Beschreibung der Hierarchie von Systemmodulen in einer Roboterarchitektur. Diese ist sehr gut auf den Planungsprozess im Rahmen einer aufgabenorientierten Programmierung übertragbar und wird daher an dieser Stelle beschrieben. Die oberste Ebene stellt eine *komplexe Fertigungsaufgabe*, wie z. B. der Montage eines gesamten Produktes aus mehreren Einzelteilen, dar. Diese Aufgabe ist in *Einzelaufgaben*, wie z. B. „Verbinde Teil A mit Teil B“, zu zerlegen, die wiederum in einzelne *Tasks*, wie „Greife Teil A“, unterteilt werden können. Die darunter liegende Ebenen sind *Elementaroperation* wie „Bewege Greifer zu Anrückframe von Teil B“, die in einzelne *Bewegungsprimitive*, die zum Beispiel die Interpolation einer Endeffektorbahn darstellen, aufgegliedert werden können. Die unterste Ebene entspricht dem unmittelbaren Roboterverhalten.

Im Vergleich der aufgabenorientierten Programmierung mit der Montageplanung wird deutlich, dass hier Überschneidungen existieren. Beispielweise stellt die Zerlegung einer komplexen Fertigungsaufgabe in Einzelaufgaben im Bereich der Montage die Generierung der Montagereihenfolge dar, die auch im Bereich der rechnergestützten Montageplanung betrachtet wird. Eine klare Trennung ist nicht immer möglich, die Ansätze lassen sich aber meist dahingehend unterscheiden, dass im Bereich der Montageplanung kein Steuerungscode für das Montagesystem erzeugt wird.

2.6 Montageplanung

Seit Mitte der 1980er Jahre wurden verschiedene Planungsmethoden für Montagesysteme entwickelt und in den 1990er Jahren in den Unternehmen umgesetzt. Die heute gängige Praxis baut noch immer auf diesen Methoden auf. Um die Anwender der Methoden bei der Planung und Optimierung zu unterstützen, wurden diese im weiteren Verlauf in digitale Werkzeuge integriert. Wegen des großen Aufwands in der Modellierung werden diese Werkzeuge aber oft noch nicht eingesetzt (NEUMANN & WESTKÄMPER 2014, S. 120).

Entsprechend gliedert sich auch dieser Abschnitt, in dem zunächst übergreifend anwendbare Entwicklungs- und Vorgehensweisen zur Montageplanung und anschließend grundlegende Konzepte der rechnergestützten Montageplanung vorgestellt werden.

2.6.1 Methoden der Montageplanung

Prinzipiell sind übergreifende Vorgehensweisen für die Planung auch für Montagesysteme einsetzbar (LOTTER & WIENDAHL 2005, S. 407). Da die Gestaltung des Montagevorgangs jedoch wenig berücksichtigt wird, sind Vorgehen, wie die VDI-Richtlinie 2221, nicht direkt für die Montageplanung geeignet (CUIPER 2000). BULLINGER (1986) schlägt eine Planungsmethodik vor, die sich aus den sechs Schritten Konzeption, Ablaufplanung, Montagesystementwurf, Ausarbeitung, Realisierung und Betrieb zusammensetzt. CUIPER (2000, S. 37) führt aus, dass BULLINGER (1986) zwar Hilfestellungen bei der Auswahl und der Gestaltung manueller und automatisierter Montagestationen gibt, jedoch nicht die Thematiken der Steuerungsplanung und der Planungsunterstützung durch durchgängige Werkzeuge betrachtet. LOTTER (1992) definiert eine Planungssystematik für Montagesysteme, welche aus insgesamt elf Schritten besteht. Die einzelnen Schritte werden jedoch abhängig vom Produkt und dessen Komplexität oder

Produktions-volumen angepasst. Im ersten und zweiten Schritt werden die Anforderungen an das Montagesystem, wie Mengengerüst und Nutzungsdauer, festgelegt bzw. bestimmt und daran anschließend das Produkt bzgl. Handhabungseigenschaften, Fügerichtungen oder Fügeverfahren analysiert. Aus diesen Grundlagen wird im dritten Schritt der Montageablauf ausgehend vom Produktaufbau und der Fügerangfolge bzw. dem Montagevorranggraph bestimmt. Die Fügerangfolge gibt an, welches Produkt vor welchem gefügt werden muss. Darauf aufbauend erfolgt die Funktionsanalyse der Einzelvorgänge, an die sich die Taktzeitermittlung, die Layoutplanung und die Personalbedarfsermittlung anschließt. In Verknüpfung mit den vier zuvor genannten Schritten erfolgt die Verfügbarkeitsermittlung des Montagesystems, aus der ein Pflichtenheft abgeleitet wird. In den letzten Schritten wird mit Hilfe der Investitionsrechnung eine Bewertung durchgeführt und die optimale Gesamtlösung ausgewählt.

Um die Montageplanung mit der Produktentwicklung zu verknüpfen, wurden ferner integrierte Ansätze entwickelt. Beispielsweise konzipiert FELDMANN (1997) aufbauend auf der Arbeit von SCHUSTER (1992) und der VDI-RICHTLINIE 2221 ein Vorgehensmodell der Montageplanung, welches die Konstruktion und die Montageplanung integriert. Weitere allgemeine Vorgehen zur Montageplanung werden u. a. bei KLUGE (2011) und CUIPER (2000) diskutiert und gegenübergestellt. Ein Überblick des Stands der Wissenschaft und Technik im Bereich der Montagesystemplanung und Gestaltung für variantenreiche Produkte wird in HU ET AL. (2011) vorgestellt.

2.6.2 Grundlagen der rechnergestützten Montageplanung

Unter der rechnergestützten Montageplanung sind Konzepte zu verstehen, welche auf digitale Werkzeuge zur Unterstützung des Planungsprozesses basieren (CUIPER 2000, S. 37). Das Datenmanagement stellt bei der Verwendung von rechnergestützten Werkzeugen zur Montageplanung den Mittelpunkt und die besondere Herausforderung dar (HOLLE 2002, S. 16; NEUMANN & WESTKÄMPER 2014, S. 120). Dabei sind computerverarbeitbare Daten der Produkte und dessen Fügeverbindungen, der späteren Montageprozesse (z. B. Stückzahlen) und der Betriebsmittel sowie Basisinformationen und übergreifende Planungsdaten (z. B. Fristen) notwendig (HOLLE 2002).

Im Bereich der rechnergestützten Montageplanung können Ansätze unterschieden werden, die sich auf den gesamten Montageplanungsprozess oder nur auf

Teilbereiche konzentrieren. Teilbereiche die besonders im Betrachtungsbereich stehen sind:

- die automatische Ableitung von Montagereihenfolgen (CUIPER 2000, S. 48; TÖNSHOFF ET AL. 1992; KAUFMAN ET AL. 1996),
- der Einsatz von Simulationswerkzeugen zur Überprüfung von Planungsergebnissen (LOFERER 2002),
- Systeme zur Auswahl und Anordnung von Betriebsmitteln (LOFERER 2002), wie sie beispielsweise von LUETH (1992) für die automatische Layoutplanung von Roboterzellen gezeigt wird,
- und Systeme zur Konfiguration der Betriebsmittel für die konkrete Montageaufgabe (LOFERER 2002).

3 Stand der Wissenschaft und Technik

Der Stand der Wissenschaft und Technik gliedert sich in vier grundsätzliche Bereiche, die sich aus der beschriebenen Zielsetzung und den Grundlagen ableiten lassen. Dies sind zunächst *digitale Modelle im Umfeld der Automatisierungstechnik*, die als Grundlage für das Informationsmodell bei der aufgabenorientierten Programmierung dienen können. Die weiteren Bestandteile dieses Kapitels gliedern sich in die Teilbereiche *rechnergestützte Montageplanung, Konzepte zur impliziten und aufgabenorientierten Programmierung* sowie *Methoden und Vorgehensmodelle zur Einführung von Softwaresystemen in Unternehmensprozesse.* Unter den *Konzepten zur impliziten und aufgabenorientierten Programmierung* werden alle Ansätze zusammengefasst, welche einer modellbasierten oder impliziten Programmierung sowohl für SPS als auch IR zugeordnet werden können.

3.1 Digitale Modelle im Umfeld der Automatisierungstechnik

Wie in Kapitel 2 ausgeführt, ist für eine aufgabenorientierte Programmierung eine Modellierung der Anlage bzw. der Ressourcen als auch der Aufgabe, d. h. der zu produzierenden Produkte und zugehörigen Prozesse, notwendig. Im Folgenden werden bestehende Konzepte, Sprachen und Standards zur Modellierung in den Bereichen Produkt, Prozess bzw. Verhalten und Ressourcen vorgestellt. Zuvor werden auch übergreifende Ansätze diskutiert, die alle drei Teilbereiche abdecken. Abschließend werden Forschungsansätze, die sich mit ausgewählten Bereichen der Modellierung befassen, analysiert. Ein Fokus liegt auf der Beschreibung der Funktionen bzw. Skills von Ressourcen.

3.1.1 Domänenübergreifende Modellierungssprachen und Standards

Die *Unified Modeling Language (UML)* ist eine grafische Modellierungssprache mit der sich statische, strukturelle und dynamische Aspekte von Software und anderen Systemen mit Hilfe von 14 verschiedenen Diagrammarten abbilden lassen (SCHÄLING 2013). Die UML eignet sich, um logische Zusammenhänge zu modellieren. Die Beschreibung physischer Eigenschaften ist jedoch nur bedingt möglich (BARBIERI ET AL. 2013, S. 2).

SysML ist eine grafische Modellierungssprache für das Systems Engineering, die auf der UML-2.0-Spezifikation basiert. Sie enthält fachgebietsspezifische Erweiterungen und verzichtet gezielt auf bestimmte Teile des UML-Standards (DELLIGATTI 2014, S. 10–13). Somit ermöglicht sie im Gegensatz zu UML eine Model-

lierung in allen Domänen (z. B. auch der Mechanik) und über alle Lebenszyklen eines Systems hinweg (WEILKIENS 2008, S. 23). Jedoch bemängeln JI ET AL. (2011, S. 9) an SysML die fehlende Möglichkeit zur Beschreibung von zeitkontinuierlichen dynamischen Modellen und Anforderungen. Diese Bereiche werden als nicht ausreichend für reale Industrieanwendungen gesehen.

AutomationML (Automation Markup Language) stellt ein in der DIN EN 62714 standardisiertes XML-basiertes Datenaustauschformat für den Engineeringprozess von Produktionssystemen dar, das die Systembrüche zwischen unterschiedlichen Engineering-Werkzeugen überwinden soll. Das vorrangige Ziel von AutomationML ist die vollständige Beschreibung kompletter Produktionsanlagen mit allen Komponenten über alle Engineering-Phasen hinweg. Dazu werden bestehende Standards wie CAEX (Computer Aided Engineering Exchange) als Dachformat und COLLADA, MathML oder PLCopen XML kombiniert. *CAEX* ist ein nach DIN EN 62424 standardisiertes XML-Datenformat, welches mit objektorientierten Grundkonzepten für die Modellierung von Anlagenstrukturen bzw. den Austausch von leittechnikrelevanten Informationen in der Verfahrens- und Leitertechnik konzipiert wurde (EPPLE 2003, DRATH & FEDAI 2004b, 2004a). AutomationML ist erweiterbar und erlaubt über Interfaces die Einbindung weiterer Standards. Die Beschreibung der für die Erstellung von Roboter und SPS-Programmen notwendigen Informationen, sowie die Speicherung der jeweiligen Programme sind darin explizit vorgesehen (DRATH 2010). Abbildung 8 zeigt die Architektur von AutomationML.

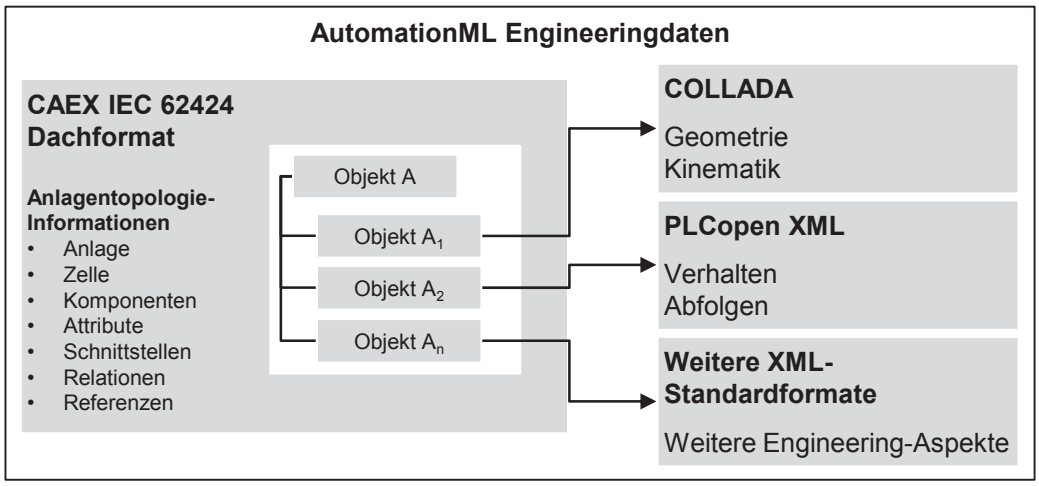


Abbildung 8: AutomationML-Architektur nach DRATH (2010)

AutomationML übernimmt aus dem CAEX-Format neben dem Objektmodell wesentliche Elemente, die nachfolgend kurz ausgeführt sind (DRATH 2010):

- Klassen, Vererbungen und Bibliotheken vereinfachen die Modellierung und ermöglichen die Wiederverwendung von Modellen.
- Die Topologie einer Anlage wird in einer Instanz-Hierarchie beschrieben, welche alle beteiligten Objekte, Attribute, Schnittstellen, Relationen und Referenzen umfasst.
- Mit Rollen ist es möglich abstrakte Klassen zu modellieren (z. B. die Rolle „Greifer“ bei einem Greifer). Rollen ermöglichen somit die semantische Identifikation von Objekten.
- Mit Schnittstellen, Relationen und Referenzen können sowohl die Verbindung von CAEX-Objekten untereinander, als auch die Verbindungen von CAEX-Objekten zu externen Dokumenten, modelliert werden.

Zur Beschreibung einer Produktionsanlage und der dort produzierten Produkte wird ein Drei-Sichten-Konzept verwendet. Dieses besteht aus den Teilbereichen Produkt, Prozess und Ressource (PPR), welche jeweils miteinander in Beziehung stehen. Für die Beschreibung von Prozessen, Logik und Verhalten stehen insgesamt sechs Diagrammtypen zur Verfügung. Es handelt sich um Gantt Charts, PERT Charts, Impuls-Diagramme, Sequential Function Charts (SFC), Zustandsdiagramme (State Charts) und Logic Networks. AutomationML verwendet zur Speicherung der logischen Elemente das offene Datenformat PLCopen XML, greift jedoch nur auf die Sprachen SFC und FBS zurück. Die anderen Beschreibungsmittel werden über eine einheitliche Zwischenschicht, den sogenannten Intermediate Modeling Layer (IML), auf PLCopen XML SFCs abgebildet. Transformationsregeln erlauben eine Konvertierung in den IML (DRATH 2010).

Die *ANSI/ISA 88* für Batch Prozesse bzw. die darauf aufbauende *ANSI/ISA 95* für diskrete und kontinuierliche Prozesse definieren Terminologien und Modelle für den Entwurf, die Steuerung und den Betrieb von Produktionsanlagen. Die wesentlichen Bestandteile der ANSI/ISA 88 sind ein Prozessmodell, bestehend aus Einzelschritten, ein physisches Modell zur Abbildung des Unternehmens und den technischen Einrichtungen sowie ein Kontrollmodell, welches aus einzelnen Rezeptprozeduren besteht. Rezepte sind Herstellungsvorschriften, die aus einer geordneten Reihenfolge von Teilprozessschritten bestehen, welche wiederum aus Funktionen aufgebaut sind. Der Standard ANSI/ISA 95 zielt mit seinen Inhalten vor allem auf die Integration der Geschäftsebene mit der Ebene der Automatisierungs-technik für ME- und ERP-Systeme ab (THIEL ET AL. 2008).

3.1.2 Modellierung von Produkten

Die Entstehung und damit auch die Repräsentation bzw. Modellierung von Produkten erfolgt heute im Normalfall in kommerziellen CAD-Tools und stellt damit zunächst die Abbildung der Geometrie dar (CUIPER 2000, S. 16). Für weitergehende Informationen kommen verschiedene Konzepte zur Anwendung, die anschließend vorgestellt werden.

Standards zur Modellierung der Geometrie und Kinematik

Für verschieden Softwaresysteme zur CAD-Konstruktion existieren meist auch herstellerspezifische Sprachen. Beispiele sind *JT (Jupiter Tesselation), IGES (Initial Graphics Exchange Specification)* oder die *Virtual Reality Modeling Language (VRML*). FRIEDEWALD ET AL. (2011) geben einen Überblick über gängige Austauschformate. Wegen der besonderen Relevanz für diese Arbeit wird nur das *COLLADA* Format kurz beschrieben. Bei *COLLADA (COLLAborative Design Activity)* handelt es sich um einen seit 2012 definierten ISO-Standard ISO/PAS 17506, der neben 3D-, Material- und Textur-Angaben unter anderem auch Informationen zu Kinematik, physikalischem Verhalten und Animationen unterstützt (BONIN ET AL. 2014, S. 6).

Standards zur Abbildung von Produktmodellen

Im Bereich der Produktmodellierung hat sich der Begriff „Produktmodell“ etabliert, der nicht nur die geometrischen Informationen sondern auch weitere Bauteilrelevante Bestandteile umfasst (CUIPER 2000, S. 16). Die Informationen in einem Produktmodell können nach DIN 6789-2 in die Bereiche Organisation, Technologie und Struktur unterteilt werden (OSTGATHE 2012, S. 117). Um neben geometrischen Informationen weitere Inhalte im Datenmodell abbilden zu können, werden häufig sogenannte Features bzw. Merkmale verwendet (CUIPER 2000, S. 16). In der Literatur finden sich unterschiedliche Bedeutungen des Begriffs Feature, wie z. B. technisches Element, Gestaltelement, Attribut oder Fertigungselement (LOFERER 2002, S. 43). Nach WEBER (1996) sind Features als informationstechnische Elemente definiert, die technische Eigenschaften bzw. Elemente sowie deren Werte und Relationen von einzelnen oder mehreren Produkten darstellen. Es existieren mehrere Systeme, die auf der Feature Technologie basieren, eine Auswahl kann in SPUR & KRAUSE (1997) nachgelesen werden. Eine Übersicht über weitere Produktmodellierungsmethoden wie die Skelettmodellierung wird in BOSSMANN (2007) beschrieben und an dieser Stelle nicht weiter ausgeführt. Im Folgenden werden nur zwei Standards näher beschrieben.

Der *STEP (Standard for the Exchange of Product Model Data)-Standard* (ISO 10303) wurde ursprünglich für den neutralen Austausch von 3D-Modellen zwischen CAD-Systemen mit ihren jeweils eigenen Datenformaten entwickelt (PRATT 2001). Neben der Geometriebeschreibung ermöglicht STEP aber auch die Bereitstellung von Modellen für den kompletten Produktlebenszyklus, wie z. B. für FEM-Simulationen und Fertigung (PRATT 2001). Die in STEP und STEP-NC verwendeten Datenmodelle basieren auf der formalen Datenmodellierungssprache EXPRESS, die auch Teil des STEP-Standards ist (RIDWAN ET AL. 2012, S. 427). Die fest definierte Semantik der Datenobjekte im Standard erlaubt eine Weiterverarbeitung der Daten nach festgelegten Regeln (LUDER ET AL. 2013, S. 1). Wegen der umfangreichen Inhalte und der unterschiedlichen Repräsentationsarten im Ursprungs-CAD-Programm und den STEP-Daten, erfolgt oftmals keine weiterführende, maschinelle Verarbeitung wie z. B. zur Kollisionserkennung (HOFFMEISTER 2013, S. 59). Die *DIN 4002-1* beschreibt Merkmale und Geltungsbereiche für den Produktdatenaustausch unter anderem in den Bereichen Mechanik und Elektrik und Elektronik. In der *DIN 4002-5* sind außerdem Einheiten für qualitative Merkmale normiert. Zur Reduzierung der Komplexität von Geschäftsprozessen existiert darüber hinaus das DIN-Merkmallexikon, welches eine branchenübergreifende Online-Produktmerkmal-Datenbank darstellt und auf der DIN 4002 basiert (MIEHE & MÜLLER 2010).

3.1.3 Modellierung von Prozessen und Verhalten

Da die Modellierung von Prozessen für verschiedenste Anwendungsfälle erfolgen kann, wurde in der Vergangenheit eine große Anzahl an Sprachen zur Prozessbeschreibung entwickelt. PAWLEWSKI (2014) benennt und vergleicht über 70 graphische und textbasierte Sprachen zur Modellierung von Produktionsprozessen. Im Folgenden werden für die Arbeit relevante Beschreibungstechniken und Beschreibungsstandards vorgestellt.

Die *PSL (Process Specification Language)* wurde als Prozessbeschreibungssprache und Ergänzung zum STEP-Format veröffentlicht. Die Modellierung erfolgt unter Verwendung der vier Klassen Aktivitäten, Aktivitäten-Eintritte, Zeitpunkte, Objekte und ihren Beziehungen untereinander (ISO 18629 2004). PSL kann für die formalisierte Modellierung von Produktionsprozessen eingesetzt werden und ermöglicht die Modellierung paralleler und unabhängiger Abläufe von Zustandsänderungen (GÖRING & FAY 2012, S. 81). *Petri-Netze* werden sehr häufig für die Modellierung von automatisierungstechnischen- und Geschäftsprozessen verwendet (CUIPER 2000, S. 18). Petri-Netze und die Netztheorie wurden entwickelt,

um den Informations- und Kontrolldatenfluss in informationsverarbeitenden Systemen nachzubilden. Damit sind Petri-Netze abstrakte Modelle dieser Informations- und Kontrolldatenflüsse (STARKE 1990, S. 13).

PLCOpen XML ist ein international akzeptiertes, frei verfügbares Austauschformat für in den Programmiersprachen der DIN EN 61131-3 beschriebenen Programme und Bibliotheksbausteine (DRATH 2010, S. 138). Das Format umfasst alle fünf spezifizierten Sprachen der DIN EN 61131-3. Ziel des Formates ist den Austausch von Programmen zwischen verschiedenen Benutzern oder Entwicklungs-umgebungen auf Basis der XML zu vereinfachen (PLCOPEN 2015). Wegen der besonderen Relevanz für diese Arbeit werden die darin enthaltenen SFC im Folgenden kurz vorgestellt. In einem SFC wird ein Ablauf mit Hilfe von Schritten bzw. Steps und Übergangsbedingungen bzw. Transitionen definiert. Ein beispielhafter SFC ist in Abbildung 9 dargestellt.

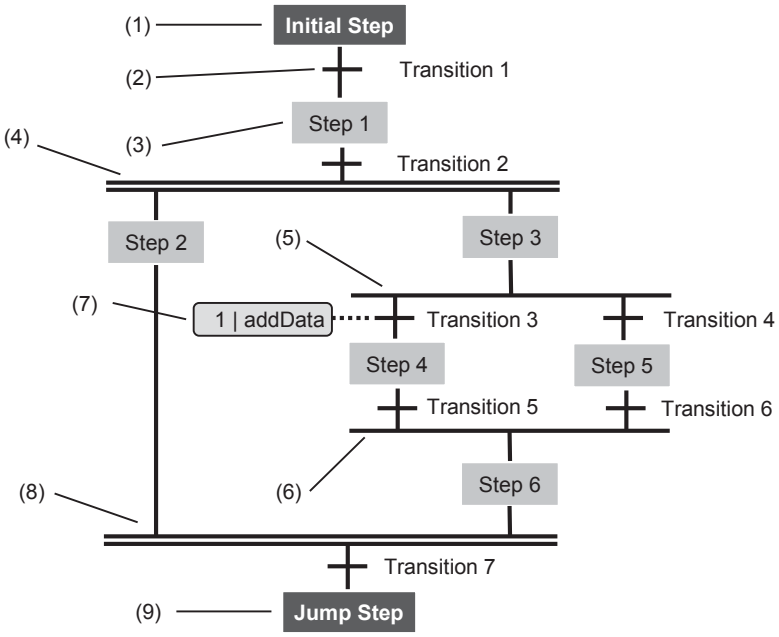


Abbildung 9: Beispielhafter SFC: (1) Initialer Schritt, (2) Transition, (3) normaler Schritt, (4) parallele Divergenz, (8) parallele Konvergenz, (5) alternative Divergenz, (6) alternative Konvergenz, (7) Datenblock und (9) Sprungschritt (in Anlehnung an DRATH (2010))

Bei den Schritten werden normale Schritte von Initial- und Sprungschritten unterschieden. Zur Modellierung existieren zusätzlich die Elemente Verzweigungen (Divergenzen) und Zusammenführungen (Konvergenzen). Es wird jeweils unterschieden nach paralleler und alternativer Verzweigung, sowie Zusammenführungen. Eine alternative Konvergenz stellt eine Verzweigung dar, in der nur

einer der nachfolgenden Kontrollflüsse gewählt wird. Mit einer parallelen Divergenz wird eine Verzweigung von parallelen Kontrollflüssen modelliert. Eine alternative Konvergenz entspricht einem logischen ODER-Operator. Zusätzliche Elemente, wie ein Aktionsblock oder Datenblock, können an Transitionen oder auch Schritten angefügt werden. Aktionsblöcke beschreiben durchzuführende Aktionen. Zusätzliche Datenblöcke werden mit „addData“-Element bezeichnet und können beliebige Informationen beinhalten.

Eine weitere relevante Möglichkeit zur Beschreibung von Prozessen wird in der VDI/VDE 3682 dargestellt. Blatt 1 und 2 beschreiben das Konzept, die graphische Darstellung sowie das zugehörige Informationsmodell. Das Informationsmodell baut auf Petri-Netzen auf und beschreibt Prozessschritte mit jeweiligen Merkmalen, denen jeweils Zustände eines Produktes vor- und nachgelagert sind. Weitere Beschreibungselemente sind Energie und technische Ressourcen. Das Informationsmodell ist übergreifend für verschiedenste Anwendungsfälle übertragbar und bietet daher eine vielversprechende Grundlage.

3.1.4 Modellierung von Betriebsmitteln

Die digitale Beschreibung von Anlagen und deren Komponenten während des Anlagenengineerings stellt aktuell eine große Herausforderung dar. Im Rahmen eines digitalen Engineerings werden für einzelne Problemstellungen wie Programmierung, CAD-Konstruktion oder Elektroplanung zumeist spezielle Tools mit den jeweiligen Datenformaten verwendet (DRATH 2010, S. 3). Zur Beschreibung von geometrischen Informationen in der Konstruktionsphase kommen z. B. CAD-Tools mit den oben beschriebenen Datenformaten zur Anwendung. Tools wie der Mechatronic Concept Designer der Firma Siemens ermöglichen darüber hinaus eine integrative Entwicklung mechatronischer Systeme, beginnend mit funktionalen Modellen und Komponentengeometrien, die mit weiteren Informationen wie kinematischen und dynamischen Eigenschaften angereichert werden (SIEMENS PLM 2014). Für die Modellierung werden jedoch herstellerspezifische Formate verwendet.

Eine Besonderheit stellen Gerätemodelle zur Beschreibung von Automatisierungs-geräten dar, welche im Folgenden kurz dargestellt werden. Gerätemodelle, d. h. Informationsmodelle für Automatisierungsgeräte, definieren Struktureigenschaften von Geräten. Die zugehörigen Gerätebeschreibungssprachen repräsentieren deren formalisierte Darstellung (SIMON 2002). Die konkreten Inhalte bzw. die Ausprägung des Gerätemodells hängen von dessen Einsatz im Lebenszyklus

der Automatisierungsanlage und dessen Einordnung in der Automatisierungspyramide ab (SELIG 2011, S. 33). SELIG (2011) stellt in seiner Arbeit verschiedene Beschreibungssprachen wie z. B. die Generic Station Description (GSD) und Generic Station Description Markup Language (GSDML) zur Beschreibung von Geräten mit Profibus bzw. Profinet-Schnittstelle vor. Dabei stellt er fest, dass diese alle für einen Spezialfall entworfen wurden und keine Darstellungsmöglichkeiten von Funktionen in den Gerätebeschreibungssprachen vorhanden sind und somit kein funktionales Engineering möglich ist. Aus diesen Beschreibungssprachen und Gerätemodellen kann zwar grundsätzlich die eigentliche Funktion eines Gerätes abgeleitet werden, diese ist jedoch kein direkter Bestandteil (NAUMANN ET AL. 2007).

3.2 Ansätze aus der Forschung zur Modellierung von Produkten, Prozessen und Betriebsmitteln

In diesem Abschnitt werden Konzepte beschrieben, die sich mit der Modellierung von Produkten, Prozessen und Ressourcen beschäftigen, jedoch nicht für die aufgabenorientierte Programmierung entworfen wurden.

3.2.1 Produktentwicklung

Um der steigenden Komplexität bei der Projektierung intelligenter Maschinen zu begegnen, entwickelte SCHAICH (2001) ein Konzept zur verhaltensorientierten Beschreibung von Produktionsmaschinen. Kernelemente sind ein objektorientiertes Referenzmodell sowie ein Konzept für dessen softwaretechnische Umsetzung. Das Referenzmodell gliedert sich in die drei Pakete Prozessbausteine, Maschinenbausteine und Endprodukte. Produktionsprozesse werden in die Typen Transformationsprozesse, Transfer- und Handhabungsprozesse sowie Kontrollprozesse unterteilt. Entsprechend dem Fokus des Ansatzes fehlt eine Verknüpfung vom Soll-Prozess mit den Funktionalitäten der Anlage. Ein weiterer Ansatz, welcher jedoch mit den gleichen Einschränkungen belegt ist, findet sich in DYLA (2002).

Zur Unterstützung des Entwicklungs- und Konstruktionsprozesses von Komponenten und Maschinenherstellern, entwickelte REUTER (2013) mechatronische Modellierungsvorgaben sowie ein Informationsmodell mit einer zugehörigen Methode zur Überführung bestehender Daten. Die Modellierungsvorgaben erlauben eine disziplinübergreifende und typenunabhängige Abbildung von Automati-

sierungskomponenten. Auch bei diesem Ansatz fehlt die Kopplung zur Steuerungsebene und übergreifende Skills werden nicht definiert.

3.2.2 Konfiguration von Produktionssystemen

Um verschiedene Automatisierungskomponenten bzgl. der durch ihre Kommunikationsobjekte zur Verfügung gestellten Funktionen vergleichbar zu machen und darin Gruppen abzugrenzen, entwickelte SELIG (2011) ein Informationsmodell zur funktionalen Typisierung von Automatisierungs-komponenten. Unter Typisierung ist die Zuordnung eines Gerätes zu einer Geräteklasse zu verstehen. Ziel der Arbeit von SELIG (2011) ist es, eine automatische Konfiguration zu ermöglichen und das Anlagenengineering zu vereinfachen. Im Rahmen seiner Analyse stellt er fest, dass die verfügbaren Gerätebeschreibungssprachen keine Darstellungsmöglichkeiten für Funktionalitäten beinhalten und entwickelt daher einen neuen Ansatz. Abbildung 10 zeigt das Konzept für die Entwicklung einer übergreifend verwendbaren, formalen, typisierten Gerätebeschreibung.

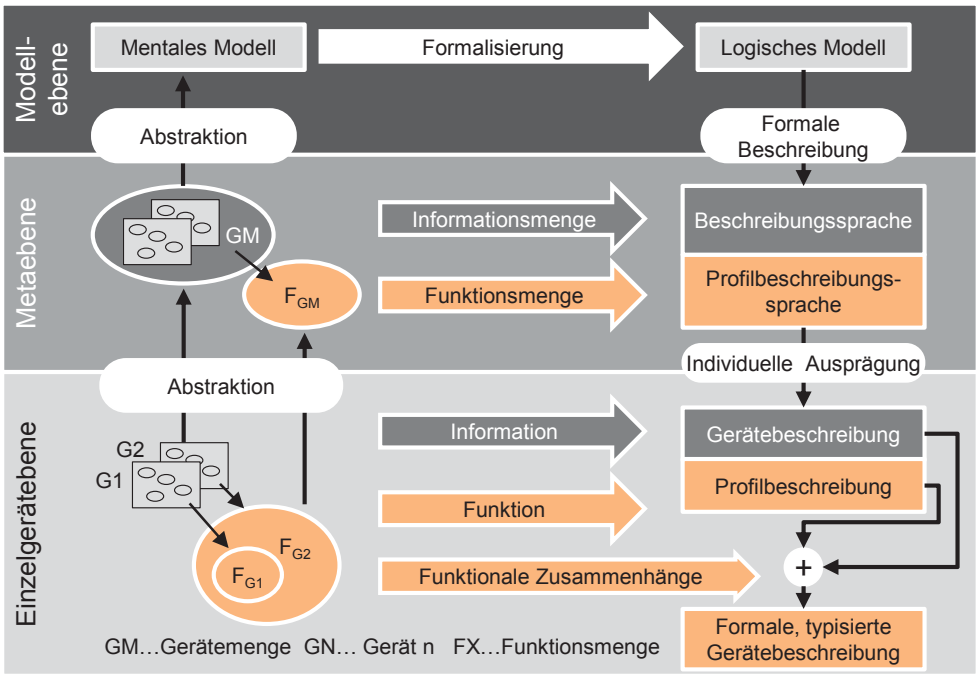


Abbildung 10: Konzept für eine formale, typisierte Gerätebeschreibung nach SELIG (2011)

Einzelgeräte werden in einer Metaebene zu Gerätemengen zusammengeführt, die entsprechend ihrer Funktionen auch eine Funktionsmenge bereitstellen. Über eine Formalisierung aus einem mentalen Modell der Metaebene der Geräte und ihrer Funktionsmengen, wird ein logisches Modell abgeleitet, welches wiederum

über eine formale Beschreibung und individuelle Ausprägung in der formalen und typisierten Gerätebeschreibung mündet (SELIG 2011).

Im logischen Modell unterscheidet er neben dem Hardwareprofil einer Ressource zwei Metaprofile von funktionalen Schnittstellen bzw. Kommunikationsobjekten, die nach dem OSI Referenzmodell strukturiert sind. Dies sind Applikations- und Kommunikationsprofile. Unter einem Profil ist die Einteilung in eine Gruppe zu verstehen und unter einem Metaprofil eine Gruppe von Profilen. Applikationsprofile werden in zwei weitere Metaprofile, Verwaltungs- bzw. generische- und applikative Funktionen eingeteilt (SELIG 2011, S. 74). Um für Kommunikationsobjekte zusammengehörige Mengen innerhalb der Metaprofile ableiten zu können, wird eine weitere Methode definiert, in welcher, basierend auf den Beziehungen der Kommunikationsobjekte, Funktionsmengen für Funktionen eines Gerätes abgeleitet und zu Gruppen zusammengefasst werden. Dazu werden u. a. folgende Eigenschaftsdimensionen zum Vergleich verwendet: Zugriffsmöglichkeiten, Änderungsmöglichkeiten, Wertebereich. Die Darstellung der Abhängigkeiten erfolgt über Graphen (SELIG 2011).

Für das Kommunikationsprofil wurden u. a. folgende Funktionsmengen unterschieden: Buskonfiguration, Kommunikationskonfiguration, Verbindungskonfiguration, Busdiagnose, Synchronisation, Überwachung, Redundanz, Hotplug, Echtzeitbits, Nichtechtzeitkommunikation und Bedarfsdaten. Als Profile auf Applikationsebene wurden folgende Gruppen identifiziert: Identifikation, Diagnose, Administration, Archivierung und Zustandsmaschine. Übergreifende Gruppen im Bereich der Applikationsfunktionen, d. h. generischen Funktionen, werden von Selig nicht vorgestellt. Des Weiteren bilden die Schnittstellen die einzige betrachtete Grundlage für die Klassifizierung von Geräten bzw. deren Funktionalitäten (SELIG 2011, S. 77).

In der Arbeit von KRUG (2013), vgl. auch in REINHART ET AL. (2010), wird eine Methode zur automatischen Konfiguration von Robotersystemen vorgestellt. Kernelement ist ein Zustandsmodell in AutomationML, welches alle für die Konfiguration relevanten Daten beinhaltet. Das Zustandsmodell wird durch einen Konfigurationsmanager automatisch aus den Gerätebeschreibungen der einzelnen Komponenten der Roboterzelle generiert und daraus die notwendigen Daten für die Zielsysteme Robotersteuerung, Programmierumgebung und Dokumentation abgeleitet. Das Konzept erlaubt durch die Verwendung von Transitionstreibern die Verwendung von Gerätebeschreibungen in beliebigen Beschreibungssprachen. Mit diesen ist eine Generierung des Zustandsmodells möglich. Die Transi-

tionstreiber bestehen aus einem Anteil, der die Dateninterpretation und einem Anteil, der den Informationskanal zur Beschaffung beschreibt. Eine Besonderheit ist, dass die Transitionstreiber nicht gerätespezifisch, sondernd an Standards und Codierungen gebunden sind. Abbildung 11 stellt die beschriebene Informationsverarbeitung in der Methode dar.

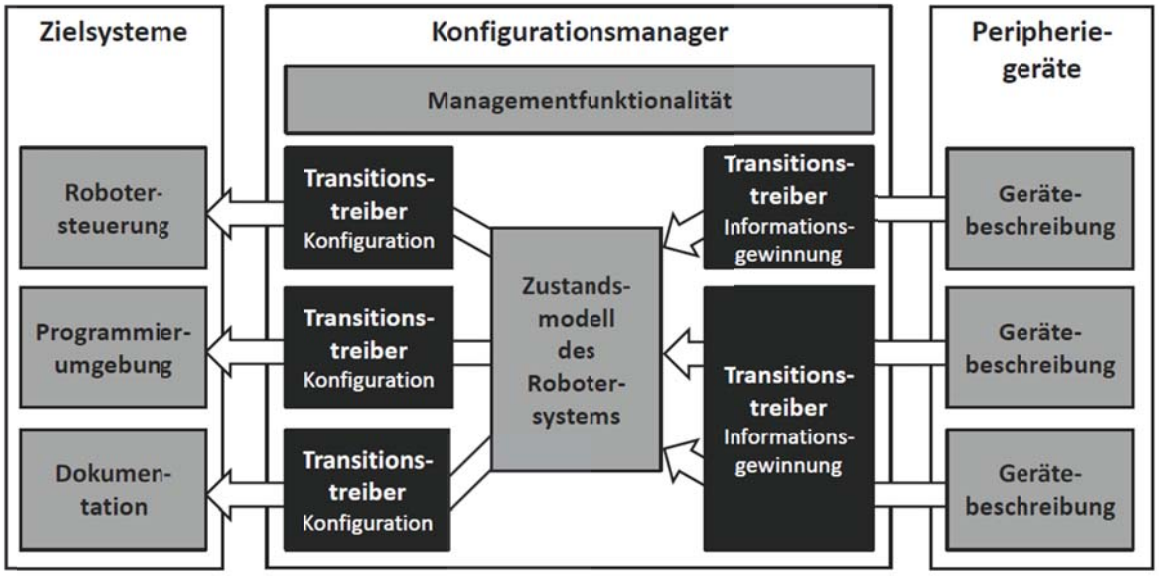


Abbildung 11: Informationsverarbeitung heterogener Datenformate und Übertragungswege während der Konfiguration (KRUG 2013)

Abbildung 12 zeigt die Architektur des Zustandsmodells, dessen Bestandteile näher erläutert werden. *Geometrische und kinematische Informationen* der Geräte werden in AutomationML integrierten Datenformat COLLADA gespeichert. Die Beschreibung der *Funktionen der Geräte* erfolgt unter Verwendung der Industrial Robot Language (IRL). Dabei handelt es sich um eine in der DIN 66312 beschriebene Norm[1], die eine herstellerneutrale Programmiersprache für Industrieroboter definiert. Zur Beschreibung des *Mapping der Prozessdaten*, d. h. der Schnittstellen von Robotersteuerung und Peripherie, werden die Schnittstellenkonzepte aus CAEX verwendet. Die Beschreibung der Informationen zur *Kommunikation* erfolgt abhängig vom Kommunikationssystem in den jeweiligen Beschreibungssprachen durch die Integration der jeweiligen Dokumente. Unterstützende Informationen wie Handbücher werden im Bereich *Hilfe* in ihren Datenformaten (z. B. pdf) integriert. *Allgemeine Informationen* wie z. B. Hersteller einzelner Geräte werden in diesem Bereich unter Verwendung von CAEX Parametern abgebildet (KRUG 2013). Zwar betrachtet KRUG (2013) auch die simula-

[1] Die entsprechende Norm wurde jedoch wieder zurückgezogen (KRUG 2013).

tionsbasierte Programmierung als Zielsystem. Die drei identifizierten Bereiche 3D-geometrische und kinematische, logische und allgemeine Informationen werden allerdings nicht ausgearbeitet und das Konzept der Skills wird nicht verwendet.

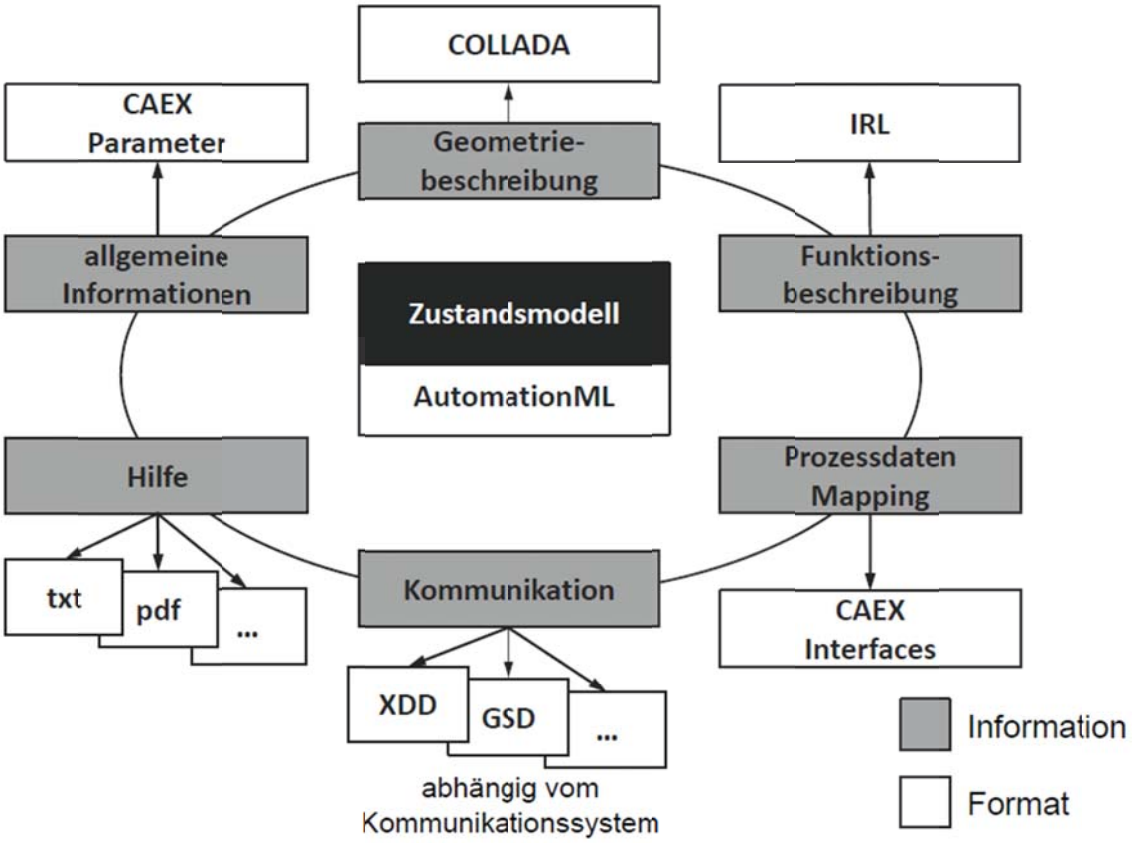


Abbildung 12: Architektur des Zustandsmodells (KRUG 2013)

HOFFMEISTER (2013) entwickelte Modelle zur aufgabengeführten Produktionsausführung. Der dezentrale Ansatz fokussiert die Teilbereiche Datenmodell, Kommunikationsmodell und Synchronisationsmodell. Eine Abstraktion der Funktionalitäten bei der Anlagenselbstbeschreibung als auch die Generierung von Steuerungscode, werden nicht betrachtet.

Aufbauend auf Software-Agenten und 'Plug+Produce'-Modulen wurde im Rahmen des Forschungsprojekts IDEAS eine 'Skill'-basierte Konfigurationsmethodik für entwicklungsfähige mechatronische Systeme erarbeitet. Das Agentenframework besteht aus mehreren Agententypen welche die Konfigurationsaufgabe durchführen können. Die in diesem Konzept verwendeten Skills werden nicht im Detail beschrieben und der Fokus liegt auf dem agentenbasierten Ansatz sowie der Konfiguration des Systems (FREI ET AL. 2009; CAVIN ET AL. 2013).

HAMMERSTINGL & REINHART (2015) stellen ein Konzept für eine einheitliche Plug&Produce-Architektur vor, mit der eine automatische Integration von Feldgeräten im industriellen Umfeld ermöglicht werden soll. Für die Integration in ein Informationsmodell werden auch Skills verwendet. Ein Treiber-basierter Ansatz ermöglicht die Heterogenität bei der Kommunikation zu kapseln. Das vorgeschlagene Informationsmodell inklusive der Skills wird jedoch nicht im Detail beschrieben.

3.2.3 Produktionsplanung

Um Produktionsaufträge auf einzelne Ressourcen bzw. Maschinen zuordnen zu können, müssen deren Eigenschaften beschrieben sein (OSTGATHE 2012, S. 10). OSTGATHE (2012) verwendet Fähigkeiten in seinem Konzept zur produktbasierten Steuerung der auftragsbezogenen Fertigung und Montage. Diese beinhalten die Funktionalitäten sowie technologische und geometrische Informationen. OSTGATHE (2012) teilt die Funktionalitäten von Ressourcen nach den in Normen (insbesondere der DIN 8580) definierten Verfahrensarten ein, die auch zur Einteilung der Prozesse in seinem Modell verwendet werden. Kinematische oder steuerungstechnische Informationen finden wegen des Anwendungsfokus keine Beachtung.

KLUGE (2011) entwickelte eine Methode zur Planung modularer Montagesysteme auf der Basis einer Fähigkeitenbeschreibung. Die Methode beinhaltet Modelle zur Beschreibung der Montageressourcen, deren Fähigkeiten, der Montageprozesse und ein Merkmalmodel für Produkte und Prozesse. Die Fähigkeiten sollen eine Verknüpfung von Prozessen und Ressourcen ermöglichen und dienen in erster Linie zur Unterscheidung der Ressourcen (KLUGE 2011, S. 77). Die Montagefähigkeiten bauen in dem Konzept auf den in Normen definierten Funktionen und Prozessen auf. Eine Verknüpfung von Fähigkeiten mit den Befehlen oder Schnittstellen von Ressourcen erfolgt entsprechend des Anwendungsfokus nicht.

Einen ähnlichen fähigkeitenbasierten Ansatz beschreibt LOSKYLL (2013). Jedoch berücksichtigen auch diese nicht die Verbindung zwischen Fähigkeiten und den Funktionen und Schnittstellen von Standardkomponenten sowie die Herleitung der Hierarchie. Im aktuellen laufenden Forschungsprojekt SkillPro sollen Skill-Beschreibungen in ein AutomationML-Modell der Systemstruktur integriert werden, um eine Planung, Kontrolle und Überwachung von Produktionssystemen zu ermöglichen (PFROMMER ET AL. 2013). Grundsätzlich setzt das Konzept auf dem Produkt-, Prozess-, Ressource-Modell aus AutomationML auf. Auch in

diesem Ansatz werden die Skills als Verbindung zwischen abstrakten Prozessen und Ressourcen verwendet. Aufgaben werden in dem Konzept als Ausführung eines Skills für ein Produkt definiert. Es ergibt sich somit ein neues Modell bestehend aus Produkt, Prozess, Ressource, Skill und Aufgabe. Die Anwendung der Modellierung sowie eine genaue Definition von Skills werden nicht behandelt.

3.2.4 Virtuelle Inbetriebnahme

LINDWORSKY (2011) stellt in seiner Arbeit eine entwicklungsbegleitende Methode zur teilautomatischen Generierung der Simulationsmodelle für die Virtuelle Inbetriebnahme vor. Die Basis hierfür ist ein sogenanntes Funktionsmodell, mit dem schon in frühen Phasen der Entwicklung das Steuerungs- und Maschinenverhalten modelliert wird und aus dem automatisch das Simulationsmodell für die VIBN und weitere Entwicklungsartefakte, wie der Elektroplan, abgeleitet wird. Einen wesentlichen Aspekt stellt die teilautomatische Erweiterung und Generierung der Verhaltensmodelle aus den erzeugten Daten des Anlagenengineerings dar. Da das vorgestellte Vorgehen für den Steuerungstest konzipiert wurde, ist die Übertragbarkeit auf die Verwendung zur Erzeugung der Modelle für die automatische Generierung des Steuerungsprogramms zu untersuchen.

BERGERT ET AL. (2010) beschreiben in ihrem Beitrag insbesondere die Modellierung von Verhalten. Zusammen bilden das Kinematik-Geometriemodell und das Verhaltensmodell das mechatronische Anlagenmodell. Diese beiden Teilmodelle müssen miteinander verknüpft werden, um z. B. eine 3D-Simulation für die Virtuelle Inbetriebnahme zu ermöglichen. Zur Beschreibung des Verhaltens werden Bewegungsgleichungen als hybride State Charts modelliert. Zusätzlich können aus dem Verhaltensmodell Vorgaben bzgl. kinematischer Ketten an das Kinematikmodell erfolgen. Die Beschreibung des Verhaltensmodells erfolgt mit PLCopen XML SFCs, wobei eine Erweiterung zur Beschreibung kontinuierlichen Verhaltens mit MathML vorgestellt wird (BERGERT ET AL. 2010). Der Beitrag zeigt die Relevanz einer Verknüpfung von Kinematik- und Geometriemodell und Verhaltensmodell, die auch auf den Anwendungsfall der aufgabenorientierten Programmierung übertragbar ist, jedoch nur einen kleinen Teilaspekt bei der Modellierung betrachtet.

Eine detaillierte Betrachtung der Verhaltensmodellierung kann außerdem in HUNDT (2012) nachgelesen werden. Im Themenfeld der Virtuellen Inbetriebnahme gibt es eine Vielzahl weiterer Ansätze, die an dieser Stelle nicht diskutiert werden (WÜNSCH 2008; DOMINKA 2007; KIEFER 2007; KUFNER 2012).

3.3 Konzepte zur rechnergestützten Montage- und Prozessplanung

In diesem Abschnitt werden Ansätze vorgestellt, die sich mit den Teilbereichen einer rechnergestützten Montageplanung, wie dem Datenmanagement oder der automatischen Generierung der Montagefolgen, beschäftigen.

Eine grundlegende Methode zur rechnergestützten Montageplanung wird von TÖNSHOFF ET AL. (1992) beschrieben. Die Methode unterteilt sich in fünf grundlegende Schritte. Im ersten Schritt werden die Stücklisten und Zeichnungen zur Beschreibung der Montageaufgabe analysiert, um daraus eine verarbeitbare Beschreibung der Montageaufgabe abzuleiten. Anschließend werden die Geometriedaten eingegeben und die Fügepositionen automatisch bestimmt. Im nächsten Schritt werden beschreibende Informationen wie Fügerichtung oder technische Funktion eingegeben. Im vorletzten Schritt wird unter Verwendung von Kriterien die Montagereihenfolge abgeleitet. Im letzten Schritt werden die passenden Fügeverfahren anhand einer Prioritätenliste ausgewählt. Die Generierung von Roboterprogrammen ist nicht Bestandteil der Arbeit.

Ein weiterer grundlegender Ansatz zur automatisierten und rechnergestützten Montageplanung ist in KAUFMAN ET AL. (1996) zu finden. Das integrierte mechanische Montageplanungssystem besteht im Wesentlichen aus zwei Bestandteilen. Ein Planungsteil, die sogenannte „Geometrie Engine“, identifiziert auf Basis von CAD-Daten und der Erreichbarkeit durch Werkzeuge einen einzelnen Plan. Der zweite Teil setzt auf der Geometrie Engine auf und betrachtet unter der Verwendung von Standardsuchalgorithmen verschiedene Montagesequenzen. Input des Systems sind die Beschreibungen der Bauteile sowie der zugehörigen Fügeverbindungen. Die Planung verfolgt die Assembly-by-Dissasembly Strategie. Das heißt, dass ausgehend vom Endzustand einzelne Teile demontiert werden und somit gültige Montagesequenzen identifiziert werden können. Der Output des Planungssystems ist u. a. die Animation des generierten Plans als auch Steuerungs-code für Roboter innerhalb der Montagezelle. Die Planung beschränkt sich jedoch auf den Prozess Zusammensetzen.

Der Sonderforschungsbereich (SFB) 336 mit dem Titel "Montageautomatisierung durch die Integration von Konstruktion und Planung" verfolgte das Ziel einer integrierten Entwicklung im Bereich der Montagesysteme (LINDEMANN ET AL. 2000). Die dafür nötigen rechnergestützten Hilfsmittel wurden im Rahmen dieses Projekts entwickelt. CUIPER (2000) teilt den Montageplanungsvorgang in die drei Schritte „Planen der Struktur“, „Planen der Teilvorgänge“ und „Planen

der Synchronisation“. Die nötige Struktur der Anlage kann aus der Struktur des Endprodukts abgeleitet werden. Lediglich die elementaren Aufgaben werden berücksichtigt (z. B. die Art des vorliegenden Fügeprozesses). Beim Planen der Teilvorgänge werden sämtliche Aspekte, wie zum Beispiel die Bahnplanung eines Roboters, festgelegt. Aus diesen ergeben sich die nötigen Betriebsmittel in der Anlage. Mit der Synchronisation der Teilschritte ergibt sich das Layout der Montageanlage, die abschließend zum Montageanlauf in Betrieb genommen wird. In der Arbeit wird die Generierung von Steuerungscode für die Steuerungen in der Montageanlage nicht im Detail betrachtet.

Um die Bewegungssequenzen von Montagevorgängen zu generieren, entwickelten HALPERIN ET AL. (2000) ein generisches Framework, den sogenannten „Bewegungs-Raum-Ansatz“. Der Bewegungs-Raum repräsentiert alle möglichen Bewegungsoperationen in Matrixform. Der Bewegungs-Raum wird aus sogenannten Blockierungs-Graphen abgeleitet, welche die Beziehungen einzelner Bauteile und deren Relationen darstellen. Durch eine mathematische Beschreibung der Zusammenhänge ist die Automatisierung in einem Algorithmus möglich. Die Verwendung der Ergebnisse des Ansatzes im weiteren Planungsprozess wird nicht betrachtet.

In SMALE & RATCHEV (2010) bzw. SMALE (2011) wird ein Skillmodell und eine zugehörige Taxonomie für die Planung bzw. Konfiguration von Montagesystemen vorgestellt. Die grundsätzliche Einteilung der Skills von Ressourcen bzw. Prozessen erfolgt nach den sechs Bereichen Bewegung, Fügen, Fixieren, Messen, Zuführen, Arbeiten. Letztere beschreibt sonstige, nicht direkt zur Montage gehörende Tätigkeiten, wie z. B. Fräsen. Die Skills werden nach vordefinierten Charakteristika wie Freiheitsgraden oder Genauigkeit weiter unterteilt. Zur weiteren Spezifikation der Ausprägung eines Skills werden jeweils qualitative und quantitative Werte verwendet. Beispielsweise wäre Traglast eine Charakteristik der Fähigkeit „Bewegen“. Die in der Taxonomie enthaltenen Skills werden der Projektbeschreibung, d. h. einem Produkt und dem zugehörigem Montageprozess und auch den Systemelementen wie einem Roboter vom Nutzer oder Systemintegrator zugewiesen. Mit den modellierten Skills erfolgt ein Abgleich und die Auswahl des passenden Equipments. Abschließend erfolgt eine Validierung und Optimierung der Auswahl. Da das Konzept auf die Auswahl von Komponenten und deren Konfiguration abzielt, beinhaltet es nicht, wie die allgemein definierten Skills mit den ansteuerbaren Funktionen von Komponenten verknüpft werden können (SMALE 2011).

MICHNIEWICZ & REINHART (2014) beschreiben ein Konzept für die automatische Analyse, Planung und Programmierung von Roboterzellen basierend auf Skills. Es wird jedoch nicht die Herleitung der Skills sowie die Verbindung zu den Funktionalitäten und Schnittstellen von Ressourcen beschrieben.

Wie in Abschnitt 2.6.2 vorgestellt, ist die Modellierung der Produkte mit Features ein häufig angewendetes Konzept. Ansätze aus dem Bereich der Montageplanung, die auf diesem Konzept aufbauen, sind nachfolgend beschrieben. ENG ET AL. (1999) stellen einen Feature-basierten Ansatz sowie ein darauf aufbauendes Konzept zur Generierung der Montagesequenz vor. Neben geometrischen Informationen werden mit Füge-Features die aus den Kontaktflächen resultierenden Freiheitsgrade in Form einer Matrix beschrieben. Die Planung des Montageablaufs erfolgt durch die Überprüfung von freien Flächen, Kollisionen sowie der Assembly-by-Dissasembly Strategie. Ein sehr ähnliches Feature-basiertes Konzept wird von MASCLE (2002) vorgestellt. Auch hier sind die wesentlichen Elemente des Bauteil-Modells ihre geometrischen Informationen sowie Features zur Beschreibung der Fügeverbindungen. Im System werden Montageoperationen in zwei Zustände, den Ausgangszustand und den Prozesszustand, der die Gegebenheiten nach der Montageoperation beschreibt, unterschieden. Prozesse werden nach zwei Arten unterschieden. Zunächst wird eine Unterscheidung gemacht, ob die Prozesse Verbindungen zwischen Bauteilen herstellen oder andere Operationen wie Reinigen durchführen. Eine weitere Unterscheidung erfolgt danach, ob der Prozess die Position der Teile verändert bzw. deformiert (z. B. Clinchen) oder ob dies nicht erfolgt (z. B. Schrauben). Das Konzept der Skills und eine Verknüpfung zu Ressourcen werden nicht verfolgt. BLEY & FRANKE (2005) beschreiben die Integration der Produktentwicklung und der Montageplanung in die digitale Fabrik. Zur Modellierung werden Features verwendet, wobei diese in die Teilbereiche Funktion bzw. Geometrie, Prozess und Qualität unterteilt sind, um damit auch komplexe Fügevorgänge wie Schweißen beschreiben zu können. Ein Schwerpunkt liegt auf der Verknüpfung des Planungsmodells mit dem Simulationsmodell zur Überprüfung der erzeugten Montagesequenzen. Die Generierung von Steuerungsprogrammen wird in dem Konzept nicht betrachtet. Eine detaillierte Darstellung des Produktmodells sowie das Vorgehen zur automatischen Ableitung einer Fügeflächenmatrix, in der die Kontaktflächen von einzelnen Produkten oder Baugruppen beschrieben sind, wird in einem weiteren Beitrag vorgestellt (BLEY & BOSSMANN 2005). Aus dieser kann wiederum ein gerichteter Fügeflächengraph abgeleitet werden, welcher in den Montagevorranggraph überführt wird. Die weitere Verwendung des Montagevorranggraphen wird nicht behandelt. In SUN ET AL. (2013) wird ein weiterer Ansatz, basierend auf

Features, beschrieben. Der Ansatz orientiert sich an den oben aufgeführten Ansätzen und beschreibt ein darauf aufsetzendes Konzept zur automatischen Generierung des Montageablaufs sowie die Umsetzung des Systems unter Verwendung kommerzieller Tools. Andere Vorhaben greifen die Feature-basierten Konzepte auf, um diese in bestehende Beschreibungsstandards zu integrieren. Beispielsweise stellen ZHA & DU (2002) ein auf dem STEP-Standard basierendes Konzept für eine integrierte Produktentwicklung vor.

Weitere aktuelle Ansätze betrachten z. B. die Planung und Optimierung des Montageablaufs unter Verwendung von genetischen Algorithmen (GALANTUCCI ET AL. 2004), neuronaler Netze (CHEN ET AL. 2010) oder Ameisen-Algorithmen und Heuristiken (WANG ET AL. 2014). Die Integration der Produktentwicklung und der Montageplanung ist auch aktuell noch Gegenstand der Forschung (DEMOLY ET AL. 2011; JONAS 2000). Weitere Untersuchungsbereiche sind z. B. die Wissensrepräsentation in der Montageplanung im Umfeld der digitalen Fabrik (RUDOLF 2007) und der Einsatz von Augmented und Virtual Reality (YE ET AL. 1999; PATRON 2004). Wegen der untergeordneten Relevanz für diese Arbeit, werden diese Bereiche nicht diskutiert.

3.4 Konzepte zur impliziten bzw. aufgabenorientierten Programmierung

In diesem Unterkapitel werden Ansätze vorgestellt, welche eine Abstraktion der Programmierung von Montagesystemen bzw. den darin vorkommenden Steuerungen ermöglichen und somit einer aufgabenorientierten Programmierung zugeordnet werden können. Wegen der großen Anzahl an Konzepten werden Beiträge vor dem Jahr 2000 nur bei besonderer Relevanz für diese Arbeit ausgeführt. Wegen der oftmals gegebenen Übertragbarkeit werden auch Konzepte vorgestellt, die sich nicht direkt auf Montagesysteme beziehen.

3.4.1 Übergreifende Programmier- und Steuerungskonzepte mit Fokus auf Industrieroboter

REINISCH (1992) entwickelte in seiner Arbeit verteilte Werkzeuge in einer CAE/CAM-Verfahrenskette inklusive deren Architektur. Für die implizite Aktionsplanung entwickelte er Verfahren zur Zerlegung abstrakter, produktnaher Fertigungsanweisungen in Aktionsprimitive. Dies beinhaltet die automatisierte Ableitung von Steuerungscode für Gerätesteuerungen. Die Aktions- bzw. Ferti-

gungsprimitive sind zum Kenntnisstand seiner Arbeit nicht bekannt (REINISCH 1992, S. 128). Diese müssen durch den Anwender definiert werden. Dabei wird eine Anlehnung an Funktion nach VDI 2680 und DIN 8593 empfohlen.

MORROW & KHOSLA (1995) bzw. MORROW & KHOSLA (1997) beschreiben ein Konzept, in dem ausgehend von Handhabungs-Primitiven zusammengesetzte Skills von Industrierobotern abgeleitet werden. Eine Klassifikation der Handhabungs-Primitive ermöglicht die Entwicklung von sensor-motorischen Primitiven, die eine Brücke von der Aufgabe zu den Fähigkeiten des Roboters darstellen. Die Handhabungsprimitive sind aus möglichen relativen Bewegungen und den damit zur Verfügung stehenden Freiheitsgraden zweier Bauteile bzw. Frames abgeleitet. Die Freiheitsgrade dienen auch als Grundlage für die Klassifikation. Die sensor-motorischen Primitive setzen sich aus den Handhabungs-Primitiven, zusätzlichen geometrischen Informationen, sowie zugehörigen Algorithmen für den Sensoreinsatz, wie beispielsweise einem Kraftsensor beim Fügen eines Stiftes in ein Loch, zusammen. Das Konzept setzt allerdings jeweils Sensoren und zugehörige Algorithmen für die Primitive voraus und ist damit nicht allgemeingültig.

HUMBURGER (1998) entwickelte ein System zur aufgabenorientierten Roboterprogrammierung. Das Hauptaugenmerk liegt auf der ganzheitlichen Optimierung von Montageprozessen. Dazu entwickelt er einen methodischen Ansatz und die zugehörige Systemarchitektur. Zur Planung bzw. Lösung von Aufgaben im Rahmen des Programmiersystems wird daher eine sogenannte Blackboard-Architektur verwendet. Im Blackboard werden von ihm drei Hierarchieebenen sowie die Einbindung in die Verfahrenskette bei der Programmierung mit STEP und IRL beschrieben. Für die Kontrolle in seiner Blackboard Architektur entwickelt er eine hierarchische Meta-Ebenen-Architektur, die in Abbildung 13 dargestellt ist. Charakteristisch für eine Meta-Ebenen-Architektur ist, dass Aktionen auf einer höheren Ebene durchgeführt werden, um die nächste Aktion auf einer niedrigeren Ebene zu bestimmen. In der dort gezeigten Architektur stellen die Wissensquellen auf unterster Ebene (Spezialistenebene) die Spezialisten für bestimmte Planungsaufgaben dar, wie z. B. die Generierung von Roboterbahnen. In der darüber liegenden Ebene erfolgt die Optimierung, die Steuerung des Planungsablaufs bzw. der Planungssequenz durch die Spezialisten sowie die Steuerung bei Problemen bzw. Sackgassen im Lösungsprozess (Aufgabenebene). Die übergeordnete Strategie steuert die Wissensquellen in der Strategieebene.

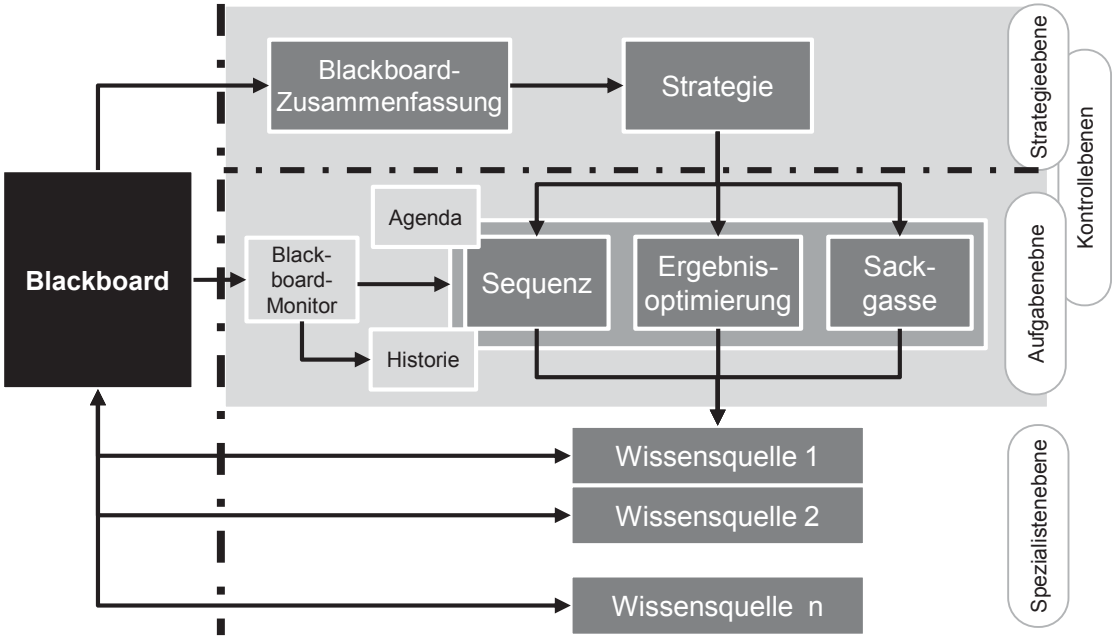


Abbildung 13: Systemarchitektur nach HUMBURGER (1998)

HUMBURGER (1998) entwickelte darüber hinaus eine dreistufige Methodik zur Definition von Wissensquellen unter Berücksichtigung von deren Abhängigkeiten und einer Zerlegung des zugehörigen Domänenwissens. Die Modellierung der Roboterzelle erfolgt durch den Nutzer. Die Modellierung der Bearbeitungsaufgabe wird nur für den Anwendungsfall des Bahnschweißens beschrieben. Ein Ansatz zur Abstraktion der Funktionalitäten von Geräten wird nicht behandelt.

Einen prozessorientierten Ansatz zur Offline-Programmierung von Industrierobotern stellt ROKOSSA (1999) vor. Ein abstrakt beschriebener Bearbeitungsprozess bildet die Grundlage zur Generierung des Steuerungsprogramms für die Ausführung des realen Fertigungsbetriebs. Der Ansatz betrachtet jedoch nicht die Abstraktion der Funktionalitäten in Montagesystemen. Aktuelle Arbeiten beschäftigen sich mit einem System zur Programmierung unter Verwendung von Funktionsmodulen, d. h. vorab erstellter und getesteter Codesequenzen (ROKOSSA 2011).

Ein System zur aufgabenorientierten Online-Programmierung von Roboterzellen (XPROB - Experimental Platform in Robotics) wird von MEYNARD (2000) vorgestellt. Abstrakte vordefinierte Befehle werden auch während der Laufzeit in ausführbare Roboterprogramme übersetzt. Das Umweltmodell wird laufend an den aktuellen Zustand angepasst, gegebenenfalls unter Verwertung von Sensordaten. Im Programmablauf wird die Aufgabenstellung im Task Interpreter zunächst syntaktisch überprüft und anschließend in einen generischen Code über-

setzt. Die Programmierung erfolgt in einer vordefinierten Sprache. Im Task Planner erfolgt die Aufteilung der Aufgabe in Einzelschritte und die Überprüfung dieser Einzelschritte auf Durchführbarkeit. Die Greifbewegungsplanung und Grobbewegungsplanung erfolgen in einem sogenannten Motion Planner. Die resultierenden Elementaranweisungen werden im letzten Schritt noch in spezifischen Robotercode umgewandelt, wobei auch Kommunikationsanweisungen erzeugt werden. Die Modellierung des Umweltmodells erfolgt in dem Konzept durch den Anwender und die Abstraktion der Funktionalitäten von Komponenten wird nur rudimentär betrachtet.

Im mehreren Arbeiten im Umfeld des SFB 562 (Robotersysteme für Handhabung und Montage) wurden an der Universität Braunschweig Konzepte für die Vereinfachung und Automatisierung der Programmierung von Industrierobotern erarbeitet. MOSEMANN (2000) beschreibt ein Konzept zur Generierung von Montageaufgaben aus montierten Baugruppen bzw. deren CAD-Daten, welches auch im SFB aufgegriffen wurde. Die identifizierten Aufgaben werden manuell in Aktionsprimitive zerlegt und deren Sequenz abgeleitet. Aktionsprimitive sind elementare Roboteroperationen, die durch Sensoren überwacht werden, eine direkte Schnittstelle zur Roboterregelung haben und durch Anfangsbedingungen und Endzustände beschrieben sind (MOSEMANN 2000, S. 72–73). Diese Operationen ändern den Zustand des Montageprozesses. Für die Ausführung der Aktionsprimitive ist jedoch eine hybride Robotersteuerung nötig (MOSEMANN & WAHL 2001). Im Rahmen des SFB 562 wurde das Konzept zu einer durchgängigen Prozesskette erweitert (THOMAS 2008; RAATZ ET AL. 2008). Der Fokus lag auf einer Weiterentwicklung zur automatischen Generierung von Aktionsprimitivnetzen aus den Baugruppenmodellen. In diesen Netzen sind die Reihenfolgen für die Montageaufgabe, die abhängig von Zugänglichkeit, mechanischer Stabilität oder Handhabbarkeit durch einen Algorithmus identifiziert werden, abgelegt (THOMAS 2008). Das Konzept hat die gleichen beschränkenden Anforderungen an den Sensoreinsatz und den Eingriff in die Robotersteuerung wie MOSEMANN (2000).

Im internationalen Forschungsprojekt IMS/HMS (Intelligent Manufacturing Systems/Holonic Manufacturing Systems) wurde ein Konzept für ein dezentrales roboterbasiertes Montagesystem entwickelt (ARAI ET AL. 2003; SUGI ET AL. 2003). Das sogenannte HAS (Holonic Assembly System) ermöglicht neben der Zerlegung der Aufgaben in Elementaranweisungen auch die Verteilung dieser Aufgaben an die jeweiligen Ressourcen (Roboter) in einem Multiagentensystem. Der beschriebene Ansatz basiert auf einem agentenbasierten Plug&Produce-

Konzept (ARAI ET AL. 2000). Hierbei werden bei der Beschreibung der Aufgabe vier Ebenen unterschieden. Der Task Layer (1) definiert die Aufgaben und besteht aus einzelnen Prozessen, dem Process Layer (2). Der Operation Layer (3) stellt eine Aufteilung des Process Layers in auf der Maschine ausführbare Elementaroperationen dar. Im Execution Layer (4) erfolgt die Verbindung mit den zugehörigen Betriebsmitteln (ARAI ET AL. 2000). Eine genaue Beschreibung der Elementaroperationen oder deren Herleitung erfolgt nicht.

Die Programmierung von Industrierobotern ohne Expertenwissen zu ermöglichen, war das Ziel des Forschungsprojektes Porthos (BRECHER ET AL. 2004). Der entwickelte Prototyp einer Programmierumgebung ermöglicht das Festlegen von Abläufen mit Hilfe von Funktionsbausteinen. In den einzelnen Bausteinen werden Roboterbefehle zu aufgabenorientierten Operationen zusammengefasst, für deren Übersetzung in das Roboterprogramm die Befehle um Parameter aus einem Datenmodell ergänzt werden. Das Konzept beinhaltet keine abstrakten Beschreibungen von Funktionalitäten und das Modell des Roboters muss durch den Nutzer erstellt werden.

Die Zielsetzung der Arbeit von LÜDEMANN-RAVIT (2005) war die Entwicklung eines Systems, zur Automatisierung der notwendigen Schritte zur Programmierung und Planung einer Roboter-Fertigungszelle unter Verwendung von CAR (Coumputer Aided Robotics). Kernaspekte des Systems sind eine roboterunabhängige Sprache zur Programmierung und eine erweiterbare Architektur der Programmierumgebung. Um eine Automatisierung der Programmierung zu erleichtern, werden von Experten vordefinierte Ablauf-Schablonen verwendet, die für die jeweiligen Anwendungsfälle parametriert werden können. Das Konzept ist damit nicht für eine weitreichende Abstraktion der Programmierung mit abstrakten Aufgaben konzipiert.

In einer Zusammenarbeit der Universität Fukuoka und der Technischen Universität Kyūshū wurde eine Methode zur Wissensrepräsentation und Programmierung von Robotermontageaufgaben mit Fokus auf Positions- und Kraftsensoren erarbeitet (ARAMAKI ET AL. 1999; ARAMAKI ET AL. 2007; NAGAI ET AL. 2007). Ausgangspunkt ist ein relationales Modell des Montageobjekts und dessen funktionale Elemente. Die Steuerungshierarchie des Konzepts ist in vier Ebenen unterteilt: Control Primitive Level (Zustandsbeschreibung einer Elementaroperation auf Sensor-Ebene), Control Skill Level (Steuerungsziele bestehend aus Control Pimitiven), Skill Level (Aktionen bestehend aus Control Skills) und Task Level (Montagesequenz bestehend aus Skills). Die Zusammenhänge der Ebenen sind

vordefiniert und erlauben eine Programmierung auf hoher Abstraktionsebene. Eine Anwendung des Konzepts mit Standardhardware und die Herleitung der Skills werden nicht beschrieben.

SECKNER (2008) stellt in seiner Arbeit ein aufgabenorientiertes System zur Programmierung von Mikromontageprozessen vor. Die Kernelemente des Konzepts sind ein Umweltmodell und ein Aufgabentransformator. Im Umweltmodell werden sowohl der Zustand des Systems (Zustandsmodell) als auch die Funktionen des Systems (Wissensbasis) gespeichert. Die Funktionen des Systems werden als Skripte, die aus Vorbedingungen, Nachbedingungen, sowie den Ablauf der Aktion bestehen, in der Wissensbasis beschrieben. Über die Skripte erfolgt eine direkte Steuerung der Mikromontageanlage. Der Aufbau des Zustandsmodells erfolgt durch den Nutzer. Die Verwendung von Standardkomponenten und Steuerungsarchitekturen wird im Rahmen der Arbeit nicht betrachtet. Die Arbeit von EHRMANN (2007) gliedert sich an den Ansatz von SECKNER (2008) an und stellt ein Konzept zur Effizienzsteigerung bei der Programmierung von roboterbasierten Montagesystemen vor. Er betrachtet jedoch auch den Bereich SPS. Im Rahmen der Arbeit wurde ein Programmiersystem entwickelt, das auf einer neu entwickelten bewegungs- und interaktionsorientierten Sprache zur Beschreibung von Prozessen sowie einem anwendungsunabhängigem Geräte- und Steuerungsmodell basiert. Die Prozessbeschreibungssprache sowie das Programmiersystem erlauben aber keine weitreichende Abstraktion der Roboteraufgaben durch den Anwender (EHRMANN 2007, S. 75).

Im Projekt SMErobot™ wurden Konzepte zur Vereinfachung der Programmierung von Industrierobotern für kleine und mittlere Unternehmen erarbeitet (SMEROBOT™ 2015). Einen Teil davon stellt eine Plug & Produce Steuerungsarchitektur für Roboterzellen dar (VERL & NAUMANN 2008). Ein Plug&Produce Modul verknüpft in dem Konzept Gerätebeschreibungen und Prozessbeschreibungen, um ausführbare allgemeine Prozess zu bestimmen. Das Modul basiert auf einer serviceorientierten Steuerungsarchitektur. Vom Nutzer vorgegebene Anwendungsbeschreibungen, durch Prozessabläufe beschriebene Zustandsgraphen, spezifizieren diese allgemeinen Prozessbeschreibungen, um ausführbaren Code zu erzeugen. Die in den Gerätebeschreibungen abstrahierten Funktionalitäten von Geräten werden nicht im Detail dargestellt oder hergeleitet. Zur Modellierung werden sowohl auf Geräte- als auch auf Prozessseite Zustandsgraphen verwendet. Prozessbeschreibungen werden von Experten definiert, Gerätebeschreibungen von Experten oder Systemintegratoren. Wie eine Optimierung und

Kontrolle der Nutzervorgaben im Rahmen des Moduls erfolgt, wird nicht ausgeführt (VERL & NAUMANN 2008).

Im aktuellen Projekt SMErobotics werden die im vorherigen Abschnitt beschriebenen Themen weiterentwickelt (SMEROBOTICS 2015). Das Konzept der modellbasierten Roboterprogrammierung soll eine kognitive Industrierobotik ermöglichen (HOLLMANN 2013). Dabei wird eine einheitliche Modellierung des Gesamtsystems mit AutomationML und dem dort integrierten Produkt, Prozess, Ressource-Konzept benutzt. Der dort vorgestellte Ansatz ergänzt das Konzept von AutomationML um den Anteil „Technologie“, die von Ressourcen bereitgestellt und von Prozessen genutzt wird. Darin sind detaillierte Informationen der Komponenten abgelegt, die die Basis zur automatischen Planung von Roboterprogrammen liefern. Die detaillierte Herleitung der möglichen Ausprägungen der „Technologie“ erfolgt jedoch nicht. Prozesse werden als ausführbare Zustandsautomaten gespeichert, die die notwendigen Planungsschritte enthalten. Das Konzept wurde für den Anwendungsfall MAG-Schweißen realisiert. Weitere Arbeiten aus dem Projekt beschäftigen sich mit der Entwicklung einer Skills-basierten Programmiersprache unter Verwendung von UML (THOMAS ET AL. 2013). Das dort verwendete Konzept der Skill-Primitive basiert auf den Arbeiten aus dem SFB 562. Mit der neuen Programmiersprache soll die Programmierung auf drei Ebenen, der Prozess-Ebene, die ein Netz von Aufgaben darstellt, der Aufgabenebene, die ein Netz von Skills darstellt und der Skill-Ebene, die ein Netz von Elementaroperationen darstellt, ermöglicht werden. Die beiden oberen Ebenen sollen die Programmierung für einfache Anwender ermöglichen. Elementaroperationen sind zu Geräten gehörige Befehle mit Rückgabewerten und Trigger-Konditionen zum Beenden der Operation. Die Herleitung von Skills und Elementaroperationen wird nicht beschrieben. Für das Konzept ist ein direkter Zugriff auf den Roboter notwendig.

Auch im Projekt SIARAS (Skill-based Inspection and Assembly for Reconfigurable Automation Systems) wurden Ansätze erarbeitet, um die Eigenschaften und Skills von Geräten in Montagesystem abstrahiert beschreiben zu können (BENGEL 2008). Ein weiterer Schwerpunkt lag auf Konzepten, um die Anforderungen aus dem Produktionsprozess mit den Skills automatisch abgleichen zu können. Die Modellierung der Produktionsaufgabe erfolgt in dem Konzept am Werkstück (BENGEL 2010). Da die relevanten Bestandteile aus dem Projekt im Wesentlichen in die Arbeit von BENGEL (2010) eingeflossen sind, wird dessen Ausarbeitung im Folgenden kurz diskutiert. Zur Modellierung von Skills wird auf bestehende Normen wie die VDI 2860 oder die DIN 8593 hingewiesen und aus den dort

verwendeten Prozessen eine Hierarchie abgeleitet. Für den Abgleich der Skills der Produktanforderungen mit den Anlagen-Skills werden die Eigenschaften abgeglichen, der genaue Vorgang aber nicht ausgeführt. Andere Arbeiten im SIARAS Projekt beschreiben einen Skill-Server, um die Aufgabenbeschreibungen des Nutzers mit den Skills von Geräten zu verknüpfen (MALEC ET AL. 2007). Dieser besteht aus zwei wesentlichen Komponenten, einer Ontologie, in der das generische Wissen des Systems abgelegt ist und einem Main-Loop, der den übergreifenden Planungsablauf beinhaltet. An diesen können auch weitere Elemente angekoppelt werden. Wie der Planungsprozess abläuft und wie das Konzept erweitert werden kann, wird nicht beschrieben. Eine Verknüpfung zu den Schnittstellen und Befehlen realer Ressourcen wird nicht diskutiert. Im ROSETTA Projekt (RObot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning) wurden die Ergebnisse aus dem SIARAS Projekt erweitert (NAUMANN ET AL. 2010). Einer der Schwerpunkte war die Programmierung von Robotern auf Aufgabenebene. Alle verfügbaren Informationen wie Gerätebeschreibungen, Planungs-Algorithmen und Fehlerstrategien, werden in ein „Knowledge Integration Framework“ eingebettet, dessen Funktionsweise und Aufbau nur sehr granular beschrieben wird. Ein weiterer Teilbereich führt die Modellierungskonzepte von MALEC ET AL. (2007) fort (BJÖRKELUND ET AL. 2011). Dabei wird die Semantik des „Knowledge Integration Framework“ zur Datenverknüpfung und -suche beschrieben und insbesondere die Modellierung von Montageaufgaben, u. a. mit Sequential Functions Charts, dargestellt (BJÖRKELUND ET AL. 2011; MALEC ET AL. 2013). Die Herleitung der Skills wird auch in diesem Projekt nicht betrachtet.

In der Arbeit von FRIEDRICH (2010) wurde ein technologieorientiertes und hybrides Programmier- und Steuerungssystem entwickelt, um eine Programmierung durch Facharbeiter ohne Programmierkenntnisse zu ermöglichen. Übergeordnete Arbeitsschritte werden im System in Elementaranweisungen zerlegt. Die Elementaranweisungen werden aber nur für den Anwendungsfall der Demontage detailliert beschrieben, nicht übergreifend abgeleitet und müssen in der Robotersteuerung integriert werden (FRIEDRICH 2010, S. 47).

HUCKABY & CHRISTENSEN (2012) stellen ein Klassifizierungs-Framework für Skills in roboterbasierten Produktionssystemen vor. Der Ansatz unterscheidet Skills und Skill-Primitve und zeigt einen Ausschnitt für deren Klassifikation im Bereich der Montage. Skill-Primitive werden mit Randbedingungen wie Zeit, Pose oder Bewegung parametriert. Deren Anwendung für eine bestimmte Aufgabe kann damit überprüft werden. Die Vorgangs-Beschreibung mithilfe von Skills

ermöglicht eine abstrakte Festlegung der Montageabläufe, unabhängig von Hardware und Implementierung, die an einem Beispiel erläutert wird. Die Unterscheidung der Klassen in der Taxonomie sowie die Verbindung der Skill-Pimitive mit den Befehlen realer Komponenten werden nicht erläutert.

An der RWTH Aachen wird derzeit an einer kognitiven Planungs- und Steuerungseinheit geforscht (BÜSCHER ET AL. 2013). Mit einem hybriden Ansatz soll die Planung und Steuerung der Montage eines Produktes, nur durch seine CAD Daten möglich werden. Kognitive Kontrolleinheiten sollen, eingebettet in ein Gesamtsystem, dieses Ziel ermöglichen. Die ausführliche Beschreibung des Planers kann in EWERT ET AL. (2010) nachgelesen werden. Der Planer leitet aus der Produktbeschreibung mit einer Assembly-by-Disassembly-Strategie einen Zustandsgraphen des Montageprozesses ab. BÜSCHER ET AL. (2013) stellen jedoch fest, dass sich die Bemühungen zur Implementierung kognitiver technischer Systeme in der Montageplanung momentan noch im Bereich der Grundlagenforschung befinden.

Im Rahmen des Forschungsprojekts TAPAS (Robotics-enabled Logistics and Assistive Services for the Transformable Factory of the Future) wird an einer Mensch-Maschine-Schnittstelle für ein aufgabenorientierten Programmiersystem geforscht. Das System basiert auf einer Drei-Ebenen-Architektur. Auf der obersten Abstraktionsebene wird die Aufgabe formuliert, z.B. „Aufnehmen des Rotors an Station A“. Die Aufgabe wird auf der darunterliegenden Ebene durch eine Sequenz von Skills dargestellt. Ein Skill beschreibt eine vordefinierte Bewegungssequenz eines Roboters. In der untersten Ebene werden die „Skills“ wiederum in ihre Elementarbewegungen die Geräte-Primitive zerlegt (SCHOU ET AL. 2013). Die Skills und Elementaroperationen sowie deren Modellierung und Abgrenzung werden in dem Konzept jedoch nicht im Detail beschrieben und können somit nicht nachvollzogen werden.

3.4.2 Übergreifende Programmier- und Steuerungskonzepte mit Fokus auf Speicherprogrammierbare Steuerungen

WITSCH & VOGEL-HEUSER (2004) stellen eine Möglichkeit der Generierung von SPS-Programmen, basierend auf einem abstrakten, systemneutralen Modell der Anlage, vor. Die Modellierung erfolgt in UML und das durch den Anwender erstellte Modell wird zur Code-Generierung auf die DIN EN 61131-3 abgebildet. Der Ansatz basiert jedoch nicht auf abstrakten Skills. Im Forschungsprojekt MODALE wurde in einem Teilprojekt die Integration von SPS und SCADA

Systemen in die Simulation von Produktionsanlagen sowie die automatische Codegenerierung erforscht (DIEDRICH ET AL. 2005). Wesentlicher Kern ist jedoch die Generierung des Steuerungsprogramms und der SCADA Konfiguration aus digitalen Planungsinformationen. Als Ausgangsprunkt für die Generierung wird ein ontologie-basiertes Referenz-Modell aus SZULMAN ET AL. (2005) verwendet, das als zentraler Datenpunkt für das Engineering von automatisierten Produktionsanlagen dient. Das Modell wird mit einem Transformator in Sequential Functions Charts übersetzt. Für einzelne Ressourcen sind die zugehörigen Funktionsbausteine vorab zu definieren. BERGERT ET AL. (2007) stellen einen Ansatz zur automatischen Codegenerierung für SPSen auf Basis digitaler Prozessinformationen vor. Verfügbare Prozesse werden in einer Bibliothek bereitgestellt. Das Konzept setzt eine Modellierung der verwendeten Ressourcen inklusive deren SPS Funktions-blöcke voraus, die mit den Prozessschritten verknüpft werden müssen. Dieser vordefinierte Code ist typischerweise schon vorhanden und getestet. Die Generierung des ressourcenspezifischen Codes ist nicht das Ziel des Ansatzes. Aus dem Prozess und Ressourcen-Modell wird abschließend über eine Transformation SPS-Code in Form von Sequential Function Charts generiert.

Unter Verwendung von STEP 7 und der 3D-Simulationsumgebung eM-PLC wurde an der Universität Uslan und dem Korea Institute of Industrial Technology ein System zur aufgabenorientieren Programmierung und Konfigurierung von SPS-Systemen entwickelt (PARK ET AL. 2008). Ausgehend von der allgemeinen Beschreibung der Montagesequenz werden ausführbare SPS-Programme generiert. Der Ansatz umfasst zudem Roboter und weitere Fördermittel. Um die aufgabenorientierte Programmierung zu ermöglichen, wird eine spezifische Struktur der SPS-Programme von Einzelmodulen vorgegeben, die einen Abgleich einer allgemeinen Steuerungssequenz auf die Komponentenfunktionen ermöglicht. Skills werden in dem Konzept nicht verwendet.

Die Generierung von SPS-Code wird auch von MÜLLER (2012) beschrieben, wobei die Modellierung auf (S-BPM) Subject-oriented Business Process Management, einer formalen Notation zum Modellieren und Ausführen von Geschäftsprozessen (FLEISCHMANN 2010) fußt. Der Fokus liegt auf einer vertikal durchgängigen Modellierung des Automatisierungssystems und der Transformation des subjekt-orientierten Modells in DIN EN 61131-3 Code. Jedoch ist in diesem Konzept eine vorherige Modellierung des Steuerungsprogramms in S-BPM notwendig. DI ORIO ET AL. (2013) beschreiben eine Methodik zur automatischen Generierung von Kontaktplänen aus einem Modell, welches aus der Gesamtheit an Spezifikationsdaten des Produktionsprozesses entsteht. Eine Planung

von einzelnen Prozessschritten, um den abstrakten Ablauf des Steuerungsprogramms zu erhalten, ist nicht integriert.

Weitere Ansätze zur Code Generierung für SPSen finden sich bei KRZYSZTOF (2005), ESTÉVEZ ET AL. (2007) oder FALKMAN ET AL. (2011). Diese weisen aber im Vergleich zu den oben genannten Ansätzen keine weiteren relevanten Elemente für diese Arbeit auf und basieren auch nicht auf dem Konzept der Skills.

3.4.3 Programmier- und Steuerungskonzepte für spezifische Anwendungsfälle

Insbesondere für die Programmierung von Robotern findet sich in der Literatur eine Vielzahl weiterer Konzepte, die oftmals einzelne Anwendungen fokussieren. An dieser Stelle wird abschließend eine kurze Übersicht dieser Konzepte gegeben.

Im Bereich der Lackiertechnik wird versucht, durch neue Ansätze zur Offline-Programmierung, den Einsatz von Robotern zu flexibilisieren sowie die erstellten Programme zu verbessern. CHEN & SHENG (2011) stellen dazu ein übergreifendes Framework vor, das eine automatische Generierung der Roboterbahnen, basierend auf CAD-Daten der Produkte und einer Modellierung der Lackierpistole als auch des Lackauftrags, ermöglicht. Einen vergleichbaren Ansatz verfolgen PICHLER ET AL. (2002) im Projekt Flex-Paint. Weitere Ansätze in dem Themengebiet finden sich z. B. bei BI & LANG (2007). Um den einfachen Einsatz von Robotern zum Schleifen und Entgraten von Gussbauteilen zu ermöglichen, beschreiben SOLVANG ET AL. (2007) einen hybriden Ansatz, der im Rahmen des Systems SAPIR (Supervised and Adaptive Programming of Industrial Robots) umgesetzt wurde. Zusätzlich zu den CAD-Daten werden auch Sensordaten und Nutzereingaben verwendet, um Roboterprogramme zu erzeugen. Für den Anwendungsfall des roboterbasierten Blechbiegens entwickelte NETO (2013) ein System zur Offline-Programmierung und Simulation von Robotern aus CAD-Zeichnungen. Einen neuen Ansatz zur Programmierung von roboterbasierten Prüfsystemen wird von TEKOUO MOUTCHIHO (2013) vorgestellt. Die Roboterbahnen zur Führung eines Laserscanners werden, basierend auf den CAD-Daten, automatisch generiert. Ebenfalls im Bereich der Prüfung existiert ein System zur automatischen Prüfung von Bohrlöchern und Innengewinden, als Weiterentwicklung des Systems aus dem Projekt Flex-Paint (BIEGELBAUER ET AL. 2004). Aus CAD-Daten werden singularitäts- und kollisionsfreie Roboterbahnen zur Führung eines Endoskops generiert. Im Rahmen seiner Dissertation beschäftigte sich

MUNZERT (2010) mit der Entwicklung von Algorithmen zur Generierung von zeitoptimalen Roboterbahnen für das Remote-Laserstrahlschweißen. Dies umfasst die notwendigen Datenmodelle zur Abbildung der Schweißaufgabe, als auch der Anlagentechnik inklusive des Industrieroboters. HATWIG (2014) entwickelte im Rahmen seiner Arbeit Bahnplanungsverfahren für verschiedene Schweiß- und Schneidprozesse wie Remote-Laserstrahlschweißen, -Wobbelschweißen, oder -Abtragsschneiden mit Industrierobotern sowie Scanneroptiken ohne Zusatzhardware. Er integrierte diese in ein aufgabenorientiertes Programmiersystem, betrachtet aber, wie auch MUNZERT (2010), nicht die Erweiterung für andere Prozesse.

3.5 Kommerzielle Programmiersysteme zur Vereinfachung der Programmierung

Auch verschiedene Anbieter kommerzieller Programmiersysteme versuchen mit ihren Produkten eine vereinfachte Programmierung von Montagesystemen und den dort eingesetzten Steuerungen zu ermöglichen. An erster Stelle sind hierbei die verschiedenen herstellerspezifischen Systeme zu nennen. Die Firmen Fanuc, KUKA, ABB und Yaskawa decken weltweit ca. zwei Drittel des gesamten Marktes für Industrieroboter ab (REUTER 2012). Beispiele für zugehörige Softwareanwendungen sind RobotStudio der Firma ABB (ABB ROBOTICS 2015), KUKA.Sim der Firma KUKA (KUKA ROBOTER 2015), ROBOGUIDE der Firma Fanuc mit verschiedenen Applikationserweiterungen z. B. für Lackierprozesse (FANUC AMERICA 2015), MotoSim der Firma Yaskawa (YASKAWA EUROPE 2015) oder Rob Office der Firma Reis Robotics (REIS 2015). Diese Offline-Programmiersysteme erlauben teilweise auch die Planung kollisionsfreier Bahnen und einen Test der erstellten Programme mit einer virtuellen Steuerung, sind aber spezifisch auf den jeweiligen Hersteller zugeschnitten. Übergreifende Simulationsanwendungen wie Process Simulate der Firma Siemens (SIEMENS PLM 2009) oder die Softwarepakete der Firma Visual Components mit den Tools 3DCreate oder 3DRealize (VISUAL COMPONENTS 2015) ermöglichen die herstellerübergreifende Simulation von Roboteranwendungen mit Funktionen zur Bahnplanung oder Erreichbarkeitsprüfung. Zur Erstellung der Simulationsmodelle werden umfangreiche Komponentenbibliotheken angeboten. Eine weitreichende Abstraktion der Programmieraufgaben ist mit diesen Systemen nicht möglich. Für einzelne roboterbasierte Anwendungen wie Schweißen oder Lackieren bieten verschiedene kleinere Hersteller Offline-Programmierumgebungen an, die eine teilweise Abstraktion und Automatisierung der Programmierung ermöglichen. So

ist mit der Anwendung RobotMotionCenter der Firma Blackbird Robotics eine einfache Bahngenerierung aus den CAD-Daten der Bauteile für verschiedene Applikationen wie Schneiden, Schweißen oder für Stoffauftrag möglich (BLACKBIRD 2015). Weitere Beispiele zur Vereinfachung der Programmierung für einzelne Anwendungen sind die Softwareanwendungen AutomAPPPS (CONVERGENT 2015), Robotmaster (Intercam SA 2015), Off-Line Painting Program (CMA ROBOTICS 2013) oder RobotWorks (KRAUSE LIB 2011). Die eben genannten Softwaresysteme ermöglichen jedoch nicht die umfassende aufgabenorientierte Programmierung für Montagesysteme und fokussieren einzelne Prozesse den Bereich der Industrieroboter.

In kommerziellen Anwendersystemen für die SPS-Programmierung wird versucht, über erweiterte Funktionalitäten wie Geräte und Datenverwaltung, Simulations-Werkzeuge, Parametrier-Editoren oder Logik-Analysatoren eine Vereinfachung der Programmierung zu ermöglichen (JOHN & TIEGELKAMP 2009). Laut einer Marktstudie aus dem Jahr 2014 dominieren die Unternehmen Siemens, Beckhoff, Rockwell, B&R, Mitsubishi und Schneider den deutschen Markt in Punkto Bekanntheit und bisherigen Verkäufen (ROTHHÖFT 2014). Für diese und weitere Steuerungen existieren zugehörige Anwendungspakete wie SIMATIC S7 der Firma Siemens (SIEMENS AG 2015), die Software RSLogix™ der Firma Rockwell (ROCKWELL AUTOMATION 2015), das Automation Studio der Firma B&R (B&R 2015), das Engineering-Framework IndraWorks der Firma Bosch Rexroth (BOSCH REXROTH 2015) oder CODESYS der Firma 3S-Smart Solutions (3S-SMART 2014), welches herstellerübergreifend angewendet werden kann. Bei allen verfügbaren Programmier- und Entwicklungsumgebungen lässt sich aber feststellen, dass keine Abstraktion der Programmieraufgaben vorgesehen ist und die Programmierung hauptsächlich in den IEC 61131-3 konformen Sprachen erfolgt. Ein Trend, der sich bei der SPS-Programmierung feststellen lässt, ist die Verwendung von Hochsprachen wie C oder C++, welche z. B. mit der Software TWINCAT 3 möglich ist (BECKHOFF AUTOMATION 2015). Anwender müssen somit nicht mehr die normierten Sprachen lernen, jedoch sind Kenntnisse in der jeweiligen Hochsprache sowie Erfahrung bzgl. der I/O-Konfiguration und anderer Spezifika bei der SPS-Konfiguration und Programmierung notwendig.

3.6 Methoden und Vorgehen zur Einführung von produktionsnahen Softwaresystemen in Unternehmensprozesse

Wie im vorherigen Abschnitt ausgeführt, betrachten bisher nur REINISCH (1992) und in Teilen HUMBURGER (1998) die Integration der aufgabenorientierten Programmierung in die Unternehmensprozesse. Jedoch liefern sie keine methodische Unterstützung für die Einführung eines aufgabenorientierten Programmiersystems. In diesem Abschnitt werden daher allgemeine Modelle, Methoden und Vorgehen zur Einführung von produktionsnahen Softwaresystemen in Unternehmensprozesse diskutiert. Die Einführung eines Softwaresystems in ein Unternehmen kann im allgemeinen wie ein Projekt behandelt werden, d. h. dass die Methoden des Projektmanagements zur Unterstützung des Prozesses angewendet werden sollten (EIGNER & STELZER 2009).

STAHN & WILMERSTAEDT (1991) beschreiben in ihrem Beitrag ein Modell, um die Einführung von rechnerunterstützten Arbeitsweisen in der Produktionsvorbereitung zu unterstützen. Ihr sechsstufiges Modell fokussiert jedoch arbeitswissenschaftliche Gestaltungsschwerpunkte. Eine Vorgehensweise zur Einführung von PDM-Systemen wird in der VDI-RICHTLINIE 2219 beschrieben. Das Vorgehen setzt sich aus den drei wesentlichen Phasen *Ist-Analyse und Soll-Konzeption, Systembewertung, Auswahl und Entscheidung* sowie *Einführung und Betrieb* zusammen. Die einzelnen Schritte werden mit unterstützenden Werkzeugen beschrieben. Im Gegensatz zu einem PDM-System handelt es sich jedoch bei einem System zur aufgabenorientierten Programmierung nicht um ein datenverwaltendes System. Die funktionalen und technischen Inhalte des Vorgehens können also nicht einfach übernommen werden. In der VDI-RICHTLINIE 4499 ist der Einführungsprozess der digitalen Fabrik beschrieben. Das beschriebene Vorgehen kann auch für die Einführung einzelner Softwaresysteme angewendet werden und setzt sich auch aus drei Phasen zusammen, die jeweils weiter untergliedert sind. Da das Vorgehen jedoch für eine Vielzahl unterschiedlicher Werkzeuge der Digitalen Fabrik angewendet werden kann, werden die Schritte im Gegensatz zur VDI-RICHTLINIE 2219 nur grob beschrieben.

Ein weiterer relevanter Ansatz ist ein generisches Vorgehensmodell zur strategieorientierten und partizipativen Einführung komplexer Softwaresysteme (KOCH 2005). Das entwickelte Phasenmodell beschreibt ähnliche Phasen wie die VDI-RICHTLINIE 2219. Der Fokus der Arbeit liegt auf der Beschreibung der organisatorischen Rahmenbedingungen, insbesondere bzgl. der Anpassung der

Geschäftsprozesse. Ein Vorgehen zur Einführung eines Systems zur Produktkonfiguration wird von GHOFFRANI (2008) beschrieben. In seinem Vorgehen baut er auf der VDI 2219 auf und erweitert diese um Aspekte, die für ein System zur Produktkonfiguration zusätzlich relevant sind. Ein weiteres bedeutsames Konzept beschreibt ein fünfstufiges Vorgehensmodell für die Einführung betrieblicher Integrationslösungen und ist in Abbildung 14 dargestellt (NEDBAL 2013). NEDBAL (2013) beschreibt für das Modell die notwendigen Rollen und Phasen mit den zugehörigen Aktivitäten, Methoden und Werkzeugen. Die Phasen seines Modells leitet er aus bestehenden Vorgehensmodellen ab, welche er mit Fallstudien anwendet, um daraus Forschungsbedarf abzuleiten. Als Integrationsdimensionen nennt NEDBAL (2013, S. 91) *Technologie, Organisation* und *betriebliches Umfeld*, als Integrationsebenen *Daten*, *betriebliche Informationssysteme, Middleware, soziale Ebene* sowie die Ebene der *Prozesse und Services*.

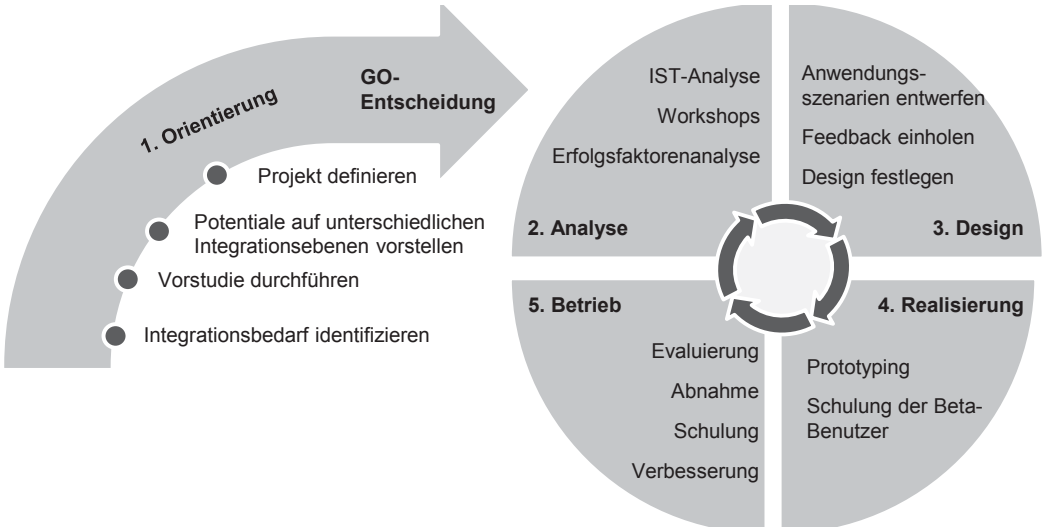


Abbildung 14: Überblick über das Vorgehensmodell nach NEDBAL (2013)

NEDBAL (2013) und KOCH (2005) geben darüber hinaus einen umfassenden Überblick zu weiteren allgemeinen Ansätzen zur Unterstützung von Einführungsprozessen. Deren Diskussion und Bestandteile können dort nachvollzogen werden. Auf Basis der Vielzahl der dort vorgestellten Ansätze kann festgestellt werden, dass eine Fülle an Methoden und Vorgehensmodellen für Einführungsprozesse im Allgemeinen und auch von Softwaresystemen existiert. Die organisatorischen bzw. projektmanagementspezifischen Inhalte und Abläufe werden darin beschrieben und sind auch zum Großteil für den Anwendungsfall dieser Arbeit übertragbar. Technische und funktionale Inhalte der Vorgehensmodelle und Methoden können jedoch nicht direkt übernommen werden und müssen erarbeitet werden.

3.7 Zusammenfassung und Handlungsbedarf

Um die im Kapitel Stand der Wissenschaft und Technik vorgestellten Ansätze mit den in Kapitel 1 definierten Zielen abgleichen zu können, werden mehrere Bewertungskriterien bzgl. der Relevanz für die aufgabenorientierte Programmierung von Montagesystemen aus der Zielsetzung abgeleitet. Diese sind:

1. **Flexibilität in der Abstraktion der Aufgabe:** Anpassbarkeit an verschiedene Anwender und Randbedingungen in Unternehmen. Liegen bspw. umfangreiche Informationen aus dem Engineeringprozess vor, so sollte der Ansatz flexibel sein, um diese zu nutzen.
2. **Flexibilität im Anwendungsbereich:** Anpassbarkeit an unterschiedliche Montageprozesse. Das jeweilige Konzept sollte nicht auf einzelne Anwendungsbereiche bzw. Prozesse, wie Schweißen, beschränkt sein.
3. **Integration in Engineeringprozess:** Bewertet, ob Vorgehensweisen oder Methoden definiert werden, um die aufgabenorientierte Programmierung in den Engineeringprozess zu integrieren.
4. **Fokus SPS und Robotersteuerung:** Wie in Kapitel 2 ausgeführt, existieren in Montagesysteme typischerweise sowohl SPS als auch Robotersteuerungen Anteilen.
5. **Standardkomponenten und Steuerungsarchitekturen:** Bewertung, ob die Ansätze von Standardkomponenten und einer Standard-Steuerungsarchitektur für die Programmierung beruhen.
6. **Informationsmodell und effiziente Modellbildung:** Bewertung, ob die Konzepte ein übergreifendes Skill-basiertes Informationsmodell für die aufgabenorientierte Programmierung beschreiben. Eine möglichst aufwandsarme, im Idealfall automatische Modellbildung zu ermöglichen, ist ein wesentliches Ziel für einen industriellen Einsatz der aufgabenorientierten Programmierung. Bestehende Informationen aus dem Engineeringprozess sollen ausdrücklich Berücksichtigung finden.
7. **Herleitung eines Fähigkeiten- bzw. Skill-Modells:** Aufgrund der besonderen Bedeutung der Skills wird bewertet, ob der jeweilige Ansatz auch die Herleitung der Skills und deren Verknüpfung mit realen Komponenten berücksichtig. Zusätzlich soll das Konzept die Programmierung adressieren.

Die im Kapitel Stand der Technik vorgestellten Ansätze wurden darüber hinaus inhaltlich den Kategorien „Modellierung“, „rechnergestützte Montageplanung“ und „implizite bzw. aufgabenorientierte Programmierung“ zugeordnet. In Tabelle 1 sind besonders relevante Ansätze aus dem Stand der Wissenschaft und Technik ausgewählt, in die inhaltlichen Bereiche eingeordnet und anhand der

definierten Kriterien bewertet. Die relevanten Eigenschaften der jeweiligen Ansätze wurden in den vorherigen Abschnitten jeweils ausgeführt und werden daher an dieser Stelle nicht im Detail behandelt.

Tabelle 1: Bewertung der Ansätze aus dem Stand der Technik und Wissenschaft

		Modellierung			Rechnergestützte Montageplanung					Implizite bzw. aufgabenorientierte Programmierung														
▬ Keine Bewertung möglich ○ Kriterium nicht erfüllt ◔ Kriterium kaum erfüllt ◑ Kriterium teilweise erfüllt ◕ Kriterium weitestgehend erfüllt ● Kriterium erfüllt		Krug 2013	Selig 2010	Kluge 2010	Cuiper 2000	Mascle 2002	Bley & Franke 2004	Smale 2011	Sun 2013	Humburger 1998	Morrow und Koshla 1999	Meynard 2000	Mosemann 2000	Seckner 2008/ Erhmann 2007	Lüdemann-Ravit 2010	SMErobot	SMErobotics	SIARAS	Rosetta	Huckaby und Christensens 2012	Friedrich 2010	Schou et al. 2013	Vogel-Heuser et al. 2005	Bergert & Dietrich 2007
1	**Flexibilität in der Abstraktion der Aufgabe**	▬	▬	▬	▬	▬	▬	▬	▬	◔	◔	◔	◔	○	○	◔	◑	◔	◔	▬	○	◔	▬	▬
2	**Flexibilität im Anwendungsbereich**	▬	▬	▬	◔	◑	◑	▬	◑	●	▬	◕	◑	◑	◕	◕	◕	◕	◕	▬	◕	◑	▬	▬
3	**Integration in Engineeringprozess**	▬	▬	▬	◑	◑	◑	◑	◑	○	○	○	○	○	◑	◔	◔	◔	◔	○	○	○	◔	◔
4	**Fokus SPS und Robotersteuerung**	◑	◑	▬	◑	◔	◔	▬	◔	○	○	○	○	●	○	○	◔	◑	◑	◔	◔	◔	○	○
5	**Standardkomponenten und Steuerungsarchitekturen**	●	●	▬	●	●	●	▬	●	●	◕	●	◑	●	●	●	●	◑	◑	●	●	●	●	●
6	**Informationsmodell und effiziente Modellbildung**	◕	▬	▬	◔	◔	◔	▬	◔	◑	◑	◑	◑	◑	◑	◑	◕	◑	◕	◑	◔	◔	◔	◔
7	**Herleitung eines Fähigkeiten- bzw. Skill-Modells**	◑	◕	◑	◔	○	○	◕	○	○	◑	◔	◑	○	○	◑	◑	◑	◑	◕	○	○	○	○

Bestehende Modellierungs- und Beschreibungsstandards wurden nicht in die Übersicht eingeordnet, da diese eher Werkzeuge zur Umsetzung einer Modellierung darstellen und somit kaum mit den Kriterien vergleichbar sind. Auch die Methoden und Vorgehensweisen zur Einführung von Softwaresystemen wurden nicht in die Tabelle übernommen, da diese nur bzgl. eines der gewählten Kriterien bewertet werden können. Da kommerzielle Programmiersysteme, wie beschrieben, nur für einzelne Roboteranwendungen die gestellten Kriterien erfüllen können, werden diese ebenfalls nicht diskutiert. Die Modellierungskonzepte und Vorgehen aus dem Bereich der Virtuellen Inbetriebnahme können

zwar teilweise für eine effiziente Modellbildung übertragen werden, betrachten aber einen anderen Anwendungsfall und sind daher auch nicht dargestellt.

Aus dem Bereich der Modellierung liefert KRUG (2013) einen Ansatz zur effizienten Modellbildung mit einem entsprechenden Informationsmodell. Dieser Ansatz kann als Basis für eine Weiterentwicklung verwendet werden.

Bestehende Ansätze zur rechnergestützten Montageplanung sehen zwar die Integration in Engineeringprozesse vor und sind flexibel in den betrachteten Prozessen, beinhalten aber bis auf eine Ausnahme (SMALE 2011) nicht die Herleitung eines Skillmodells. In diesen Ansätzen werden jedoch verschiedene Konzepte beschrieben, z. B. wie Montagereihenfolgen automatisch aus Produktdaten abgeleitet werden können (MASCLE 2002).

Im Vergleich der verschiedenen Ansätze zur impliziten bzw. aufgabenorientierten Programmierung fällt auf, dass in nahezu allen Ansätzen von einem vordefinierten Informationsinhalt der Aufgabe ausgegangen wird. HUMBURGER (1998) beschreibt ein sehr flexibles Konzept, welches sich als Ausgangsbasis für den Entwurf eines adaptierbaren, aufgabenorientierten Programmiersystems eignet. Des Weiteren wird der Integration in den Engineeringprozess sowie der Verwendung von Skills und deren Herleitung nur von wenigen Konzepten Beachtung geschenkt. Fast alle Konzepte sind darüber hinaus entweder für die Generierung von SPS- oder Robotersteuerungsprogrammen ausgelegt und erfüllen das Kriterium daher nur unzureichend. Sie beachten somit nicht die typische Architektur von Montagesystemen.

Bis auf drei Ausnahmen (SMALE 2011; SELIG 2011; HUCKABY & CHRISTENSEN 2012) betrachtet die Literatur die Herleitung der abstrakten Funktionalitäten bzw. Skills nicht und verweist auf bestehende Normen (KLUGE 2011; BENGEL 2010). Dies gilt insbesondere für Konzepte zur aufgabenorientierten Programmierung. SMALE (2011) beschreibt die Herleitung und den Aufbau von Skills am umfangreichsten, betrachtet dies jedoch für den Anwendungsfall der Planung und Konfiguration. Ein Konzept darüber, wie die Skills den Befehlen und Schnittstellen von Ressourcen zugeordnet werden können, fehlt durchgängig, ist aber zwingend notwendig für den Anwendungsfall der Programmierung. SELIG (2011) fokussiert sich in seiner Betrachtung auf Feldgeräte und berücksichtig die eigentlichen Funktionalitäten für den Produktionsprozess nicht. Für den Bereich der Verwaltungsfunktionen, liefert er eine vielversprechende Grundlage. HUCKABY & CHRISTENSEN (2012) stellen zwar auch eine umfang-

reiche Taxonomie vor, beschreiben aber nicht deren Entstehung sowie die Verknüpfung zu Ressourcen.

Zusammenfassend kann festgestellt werden, dass bisher kein Ansatz existiert, der die aufgestellten Kriterien umfassend erfüllt. Insbesondere die Herleitung eines Skillmodells, die Integration in den Engineeringprozess, die damit zusammenhängende effiziente Modellbildung mit zugehörigem Informationsmodell und die Flexibilität in der Beschreibung der Aufgabe wurden in einem durchgängigen Konzept zur aufgabenorientierten Programmierung bisher nicht betrachtet. Es bestehen jedoch Konzepte von KRUG (2013), SELIG (2011) oder HUMBURGER (1998), die in das Gesamtkonzept dieser Arbeit eingeordnet werden können. Wie gezeigt, findet sich in der Literatur eine Vielzahl von Konzepten für die Umsetzung von Planungsfunktionalitäten, wie z. B. der automatischen Ableitung einer Fügefolge, der Planung einzelner Fügeprozesse wie Schweißen oder der Planung kollisionsfreier Roboterbewegungen. Ziel dieser Arbeit ist daher nicht die Entwicklung dieser prozess- und komponentenspezifischen Planungsfunktionen. Bestehende Ansätze werden, wenn möglich, in das Konzept eingebettet.

4 Anforderungsanalyse an die aufgabenorientierte Programmierung im industriellen Umfeld

4.1 Vorgehen zur Anforderungsanalyse

Wie in Kapitel 2 ausgeführt, ist die Programmierung von Montagesystemen im Spannungsfeld zwischen den in der Montageplanung und dem Engineeringprozess erzeugten Ergebnissen und den dabei angewendeten Werkzeugen zu sehen. Die Anforderungsanalyse besteht daher aus vier Teilen. Zum einen wurde der Engineeringprozess bzw. die Montageplanung mit den verwendeten Werkzeugen, den erzeugten Daten sowie Teilschritten analysiert. Zum anderen wurde das Vorgehen zur Programmierung von Robotern und SPSen allgemein sowie für ausgewählte Anwendungsfälle untersucht. Mit der Analyse des Vorgehens bei der Programmierung sollen Unterschiede bzgl. der Steuerungssysteme und strukturelle Anforderungen an das adaptierbare Programmiersystem herausgearbeitet werden. Im letzten Schritt wurde aus den durchgeführten Analysen, die sich daraus ergebenden Anforderungen abgeleitet.

4.2 Analyse des Engineeringprozesses und der Montageplanung

Wie in Kapitel 2 definiert, beinhalten die Montageplanung bzw. die rechnergestützte Montageplanung und der Engineeringprozess eines Montagesystems überschneidende Tätigkeiten. Die typischen Teilschritte in der Montageplanung und rechnergestützten Montageplanung sowie die dort typischerweise erzeugten Daten wurden in Kapitel 2 erläutert und werden daher an dieser Stelle nicht nochmals dargelegt. Die Engineeringprozesse bzw. Montageplanungsprozesse in KMUs und bei größeren Unternehmen, z. B. im Bereich der Automobilproduktion, unterscheiden sich jedoch deutlich bzgl. der Organisation und den verwendeten IT-Werkzeugen. Im Folgenden werden diese daher getrennt diskutiert, um daraus übergreifend gültige Anforderungen abzuleiten.

4.2.1 Engineeringprozess von Montagesystemen in Großunternehmen

Für die Anforderungsanalyse wird im Folgenden eine Ist-Analyse des derzeitigen Anlagen-Engineeringprozesses für automatisierte Montagesysteme im Automobilbau nach DRESCHER ET AL. (2013) verwendet, da diese auch einzelne Tools für

die Schritte im Engineeringprozess benennt. Abbildung 15 zeigt die wesentlichen Phasen und Meilensteine. Weitere Referenzprozesse werden z. B. in WESTKÄMPER ET AL. (2013) beschrieben.

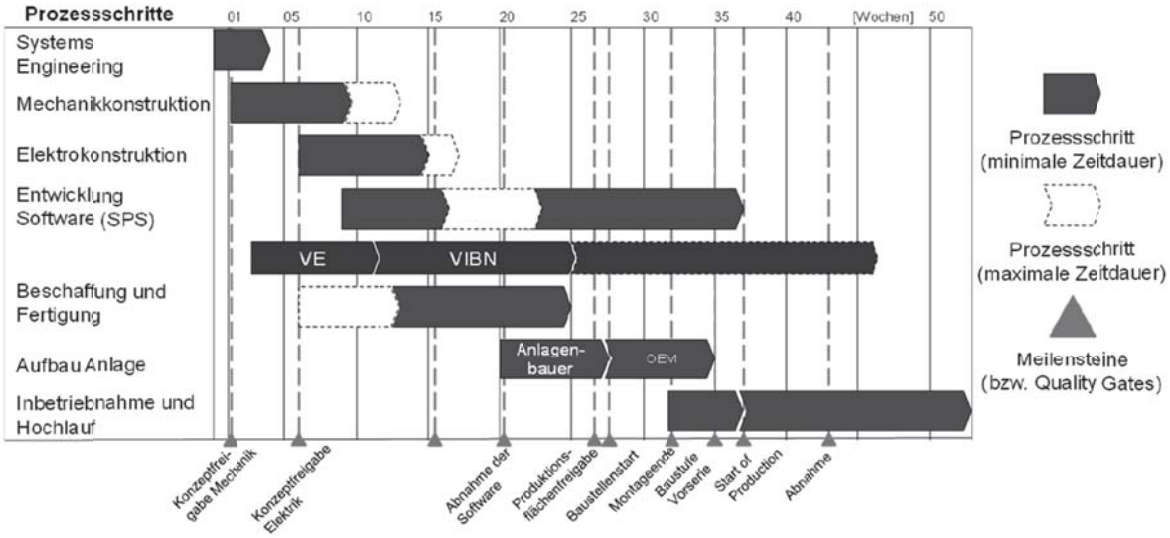


Abbildung 15: Phasen und Meilensteine im heutigen Anlagenentwicklungsprozess automatisierter Montageanlagen (DRESCHER ET AL. 2013)

Die von DRESCHER ET AL. (2013) verwendete Analyse erfolgte nach dem Konzept der Structured Analysis and Design Technique (SADT). Dabei werden einzelne Prozesselemente mit deren Eingängen, Ausgängen, Vorgaben und verwendeten Werkzeugen beschrieben. Die Abbildung 16 zeigt den SADT für die Entwicklung der Software.

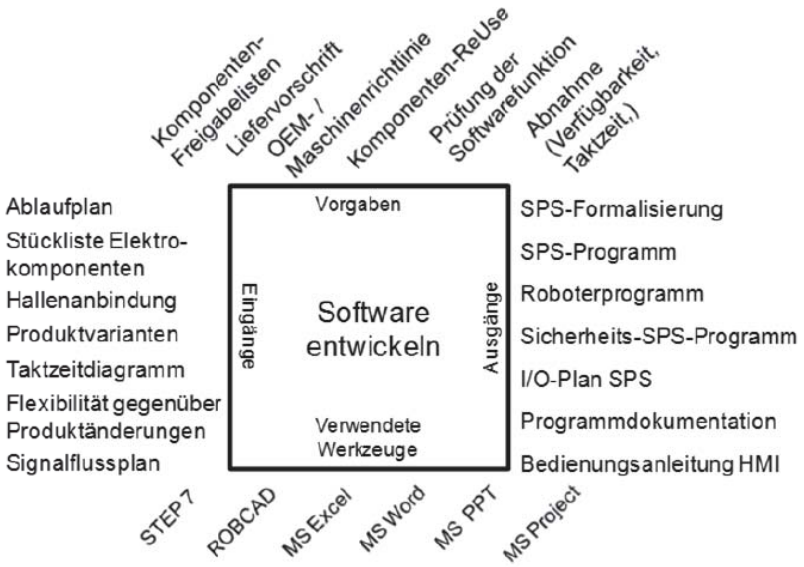


Abbildung 16: SADT für den Prozessschritt Software entwickeln (DRESCHER ET AL. 2013)

Für die Programmierung bzw. Entwicklung der Software wichtige Informationen, wie die Fügefolge oder die Produktdaten, sind auch für die Konstruktion der Mechanik, die typischerweise vor der Programmierung stattfindet, notwendige Eingangsgrößen. Die Ergebnisse der Konstruktion der Mechanik wie Feinlayout, 3D-Daten und Ablaufplan stellen wichtige Eingangsgrößen für die Programmierung dar. Auch Ausgangsgrößen des Prozessschritts „Elektrik konstruieren“ wie der Signalflussplan sind notwendige Eingangsgrößen für die Programmierung (DRESCHER ET AL. 2013). Es kann also festgestellt werden, dass eine Vielzahl an Informationen für die Programmierung in vorhergehenden Schritten generiert werden. Insbesondere die Montagereihenfolge und notwendige Sekundärprozesse sind somit vor der Programmierung durch die Anlagenarchitektur festgelegt. Diese müssen also nicht zwingend durch das Programmiersystem generiert werden.

4.2.2 Montageplanungsprozesse in KMU

Die Montageplanung in KMU ist oftmals von manuellen Tätigkeiten geprägt. Mitarbeiter übernehmen vielfältige Aufgaben und der Einsatz digitaler Werkzeuge erfolgt nur in Teilbereichen (SCHACK 2008; WILDEMANN 2010). Durch die kleinen Mitarbeiterzahlen sowie die geringe Umsatzstärke können vor allem personelle und finanzielle Ressourcen als Probleme angesehen werden (DOMBROWSKI & SCHMIDTCHEN 2010), die auch für den Einsatz der aufgabenorientierten Programmierung eine wichtige Randbedingung darstellen. Als Herausforderungen der Montageplanung bei KMU lassen sich folgende Punkte herausstellen (WILDEMANN 2010):

- wenig strukturierte Vorgehensweise bei der Montageplanung
- kaum Einsatz von Simulationssystem zur Montageplanung
- Montagevorranggraphen werden nur wenig verwendet
- hohe Flexibilitäts- und Variantenanforderungen

Die genannten Randbedingungen müssen beim Entwurf eines aufgabenorientierten Programmiersystems, welches auch bei KMU zum Einsatz kommen soll, Beachtung finden. Die genauen Datenflüsse lassen sich wegen der großen Unterschiede nicht allgemeingültig herausarbeiten. Damit kann gefolgert werden, dass die Flexibilität in der Abstraktion der Aufgabe für KMU besondere Relevanz hat.

4.3 Analyse des Vorgehens zur Offline-Roboterprogrammierung

Die Analyse in dieser Arbeit lehnt sich an das Vorgehen von KRUG (2013) an. KRUG (2013) wählt bei der Anforderungsanalyse seiner Arbeit für die automatische Konfiguration von Robotersystem zwei Anwendungsszenarien aus den Bereichen Handhaben und Schweißen, um aus diesen die Informationsstruktur ableiten zu können. Auf die Programmierung geht er jedoch nicht im Detail ein. Aus dem in der Literatur beschriebenen allgemeinen Vorgehen der Offline-Roboterprogrammierung (1) und der Betrachtung repräsentativer Anwendungsfälle (2) wird ein übergreifendes Vorgehen zur Offline-Programmierung von Industrierobotern abgeleitet (3).

Im Rahmen dieser Arbeit wird davon ausgegangen, dass die Konfiguration des Robotersystems schon erfolgt ist und die zugehörigen Daten, wie das Mapping der Ein- und Ausgänge, zur Verfügung stehen. Zur Aufnahme des Programmierprozesses und der darin vorkommenden Aktivitäten und Datenobjekte wurden erweiterte ereignisgesteuerte Prozessketten (eEPK) verwendet. Bei diesen handelt es sich um eine Modellierungsmethode für Geschäftsprozesse zum Zweck deren informationstechnischen Realisierung auf Basis der UML (SCHEER 2001). Folgende Elemente wurden aufgenommen: Ereignisse, Aktivitäten, Prozessschnittstellen, Datenformate, Informationsquellen und -ziele.

(1) Allgemeines Vorgehen der Offline-Roboterprogrammierung

Die aufgabenorientierte Programmierung kann der Offline-Programmierung zugeordnet werden. Diese wird typischerweise durch Simulationsprogramme unterstützt. PAN ET AL. (2010) geben einen Überblick des eingesetzten Vorgehens bei der Offline-Programmierung von Industrierobotern, welches als Ausgangsbasis verwendet wurde. Im Rahmen der Anforderungsanalyse wurden in dieser Arbeit fehlende Aspekte, vor allem aus dem Bereich Simulation, nach der Beschreibung von VOGL (2009) ergänzt.

(2) Betrachtung ausgewählter Anwendungsfälle

Im Rahmen dieser Arbeit wurden in Anlehnung an KRUG (2013) zwei Szenarien aus den Bereichen Handhaben und Schweißen verwendet, um das allgemeine Vorgehen zur Programmierung zu detaillieren. Die Roboterzelle für das Anwendungsbeispiel Handhaben besteht aus einem Industrieroboter, einem pneumatischen Zweibackengreifer, einem Kamerasystem, einem umlaufenden Förderband sowie einer hydraulischen Presse. Die exakte Position des Bauteils auf dem För-

derband soll durch das Kamerasystem detektiert, das Bauteil durch den am Roboter angebrachten Greifer aufgenommen, bewegt und in die hydraulische Presse eingelegt werden. Im Anwendungsbeispiel Schweißen besteht die Roboterzelle aus Industrieroboter, Schweißgerät und Schweißbrenner sowie einer Spannvorrichtung, welche auf einem Positioniertisch befestigt ist. Im Szenario sollen zwei Bauteile mit einer Kehlnaht verschweißt werden. Für beide Anwendungsfälle wurde die Programmierung beispielhaft durchgeführt. Wie auch in (1) wurde in diesem Schritt eine simulationsunterstütze Offline-Programmierung angewendet.

(3) Ableitung eines übergreifenden Vorgehens zur Offline-Programmierung von Industrierobotern

Die in (1) und (2) erarbeiteten Informationen wurden anschließend in ein übergreifendes Vorgehen zur simulationsunterstützen Offline-Programmierung von Industrierobotern in Form eines eEPKs zusammengefasst. Das Ergebnis inklusive der zugehörigen Tätigkeiten und erzeugten Daten ist wegen dessen Umfang an dieser Stelle nicht dargestellt und kann in Anhang 1 nachvollzogen werden.

Die wesentlichen Aspekte, welche in die Anforderungen mit einfließen, werden an dieser Stelle kurz diskutiert. Die Teilbereiche des Vorgehens zur Programmierung sind die Modellierung der Roboterzelle, die Bewegungsplanung, die Prozessplanung, das Post-Processing sowie die Simulation und Kalibrierung. Diese Teilbereiche müssen also auch in einem aufgabenorientierten Programmiersystem abgebildet werden. Die Bewegungsplanung stellt den wichtigsten Teilbereich des Vorgehens dar. Entsprechend sind auch die zugehörigen Informationen wie Stützpunkte oder Geometrieinformationen besonders relevant.

4.4 Analyse des Vorgehens zur Programmierung einer SPS

Das bei der Analyse der Roboterprogrammierung verwendete Vorgehen wurde auch bei der Analyse der Programmierung einer SPS zur Steuerung eines Montagesystems gewählt. Neben der Auswahl eines Anwendungsfalls und der Prozessaufnahme für diesen Anwendungsfall, erfolgt ebenfalls die Analyse des allgemeinen Vorgehens bei der Programmierung einer SPS aus der Literatur. Für den Anwendungsfall der SPS-Programmierung beinhaltet die Analyse des Prozesses auch die Konfiguration der SPS, da diese in enger Abhängigkeit von der Erstellung des Steuerungsprogramms steht. In der späteren Arbeit soll die Konfiguration jedoch nicht betrachtet werden. Die notwendigen Informationen und Aktivitäten werden wieder mit Hilfe der eEPK protokolliert.

(1) Allgemeines Vorgehen bei der Programmierung einer SPS

Zur Aufnahme des typischen Vorgehens bei der SPS Programmierung wird auf die Beschreibungen in der Literatur zurückgegriffen (LUCAS & TILBURY 2003; SIEMENS AG 2006; LJUNGKRANTZ & AKESSON 2007; JOHN & TIEGELKAMP 2009). Die dort beschriebenen allgemeinen Schritte und notwendigen Eingangsgrößen unterscheiden nicht die verwendete Programmiersprache. Die Vorgehensschritte und Eingangsgrößen sind damit unabhängig vom Anwendungsfall.

(2) Betrachtung eines ausgewählten Anwendungsfalls

Für die Steuerung mit einer SPS wurde ein Anwendungsfall gewählt, der eine Vielzahl typischer Montagekomponenten abdeckt. Als Anwendungsfall diente ein Teilmodul einer miniaturisierten Modellfabrik, die am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) aufgebaut wurde. Dieses setzt den Montageprozess „Zusammensetzen“ um, indem eine leere Papierschachtel auf eine befüllte Schachtel gesetzt werden soll. Das Modul besteht neben der Steuerung aus den Teilkomponenten Förderband, Handlingsystem mit Sauggreifer, Schieber sowie einem Trichter und Gegenhalter zum Fügen inklusive Sensoren zur Detektion der Anwesenheit.

(3) Ableitung des übergreifenden Vorgehens zur Programmierung einer SPS

Im letzten Schritt wird wieder das übergreifende Vorgehen aus dem allgemeinen und spezifischen Teil abgeleitet und mit einer eEPK beschrieben. Das Ergebnis inklusive der zugehörigen Tätigkeiten und erzeugten Daten kann in Anhang 1 nachvollzogen werden.

Abschließend sollen zunächst die wesentlichen Bereiche bei der Programmierung einer SPS beschrieben werden, welche in die Anforderungen mit einfließen. Danach erfolgt eine Beschreibung der Unterschiede und Gemeinsamkeiten bei SPS und Roboterprogrammierung. Wesentliche Teilbereiche des Vorgehens bei der Programmierung einer SPS sind die Definition der Steuerungsaufgabe, die Entwicklung des Steuerungsprogramms, das Festlegen der Laufzeiteigenschaften sowie das Testen und Verbessern. Auffällig ist, dass oft Standardfunktionsbausteine und vorhandene Funktionsbausteine verwendet werden.

Im Vergleich mit dem Vorgehen zur Offline-Roboterprogrammierung fällt auf, dass sich die Schritte nur teilweise überschneiden. Um für beide Steuerungsumgebungen Programme erzeugen zu können, muss das Programmiersystem eine große Flexibilität aufweisen. Darüber hinaus ist auffällig, dass die Funktionalität

des SPS-gesteuerten Systems schon weitestgehend durch die Mechanik der Anlage vorgegeben ist und damit eine geringere Flexibilität im Vergleich zum roboterbasierten System vorhanden ist. Da die Funktionalität des SPS-gesteuerten Moduls auch durch sehr unterschiedliche Bestandteile dargestellt werden kann, z. B. bzgl. der Anzahl der verwendeten Aktoren, besteht hier auch ein wichtiger Unterschied im Vergleich zu einem roboterbasierten System. Wegen der größeren Heterogenität der Bestandteile ist von einem niedrigeren Abstraktionsniveau der Aufgabe auszugehen. Das bewegungsorientierte Vorgehen bei der Roboterprogrammierung findet sich außerdem nicht bei der SPS-Programmierung. Auch das Festlegen der Laufzeiteigenschaften ist bei der Roboterprogrammierung im Gegensatz zur SPS nicht relevant. Überschneidungen gibt es z. B. bei der Zerlegung der Aufgabe in Teilschritte, welche bei beiden Steuerungsplattformen durchgeführt wird.

4.5 Anforderungen an ein adaptierbares aufgabenorientiertes Programmiersystem und die zugehörige Modellierung

4.5.1 Zielkriterien für den Einsatz der aufgabenorientierten Programmierung

KRUG (2013) definiert drei übergreifende Zielkriterien für den Einsatz von Plug&Produce Robotersystemen. Diese lassen sich im Grundsatz auf den Anwendungsfall der aufgabenorientierten Programmierung übertragen und stehen zum Teil im Gegensatz zueinander. Aus den Zielkriterien lassen sich die detaillierten Anforderungen ableiten.

Effizienz: Für den Einsatz der aufgabenorientierten Programmierung ist deren Effizienz von entscheidender Bedeutung. Die notwendigen Aufwände, z. B. zur Modellierung, dürfen den Nutzen, d. h. die Verkürzung des Programmierprozesses, nicht überschreiten. Diese allgemeine Anforderung wird in den nachfolgenden technischen Anforderungen weiter detailliert.

Universalität und Flexibilität: Die zu entwickelnden Methoden, Modellierungskonzepte sowie das Programmiersystem sind so zu entwerfen, dass sich diese einfach an spezifische Randbedingungen, wie Unternehmensprozesse, Anlagenkomponenten oder Nutzer, anpassen lassen.

Transparenz: Die Integration des Programmiersystems in die Unternehmensprozesse, als auch die Bedienung des Programmiersystems, müssen klar verständlich und einfach für den Anwender sein.

4.5.2 Methodische und organisatorische Anforderungen

Wegen der Heterogenität der Engineering- bzw. Montageplanungsprozesse und der dort verwendeten Tools, muss eine Methode oder ein Vorgehensmodell entwickelt werden, mit der bzw. dem die geforderte Universalität und Flexibilität des aufgabenorientierten Programmiersystems genutzt und dieses passgenau in die Unternehmensprozesse integriert werden kann. Die Verknüpfung mit den im Engineeringprozess verwendeten IT-Werkzeugen ist vorzusehen. Der Fokus soll auf den spezifischen Randbedingungen durch eine aufgabenorientierte Programmierung liegen. Wie in Abschnitt 3.6 beschrieben, existieren in der Literatur eine Reihe von Modellen und Vorgehen für die allgemeine Einführung von Softwaresystemen in Unternehmen. Ein rahmengebendes Vorgehensmodell muss die nachfolgenden Anforderungen erfüllen:

Ein aufgabenorientiertes Programmiersystem ist, der Einordnung nach NEDBAL (2013) folgend, primär in der Ebene *betriebliche Informationssysteme* in den Teilbereich „fachspezifische Anwendungen“ einzuordnen, zu dem z. B. auch CAD-Systeme gehören. Eingeordnet in die Werkzeuge der digitalen Fabrik nach BRACHT ET AL. (2009) handelt es sich, um ein *Erzeuger- oder Autorensystem*. Das Vorgehensmodell bzw. die Methode sollte also diesem Bereich zugeordnet werden. Für den Anwendungsfall dieser Arbeit ist es relevant, dass die zu verwendende Basis ein *Rahmenwerk für Organisation und Projektmanagement bei der Einführung* darstellt und *Aktivitäten, Methoden* und *Werkzeuge* beinhaltet.

4.5.3 Strukturelle und modellierungstechnische Anforderungen

Die technischen Anforderungen lassen sich in strukturelle sowie modellierungstechnische Anforderungen unterteilen.

Strukturelle Anforderungen

Betrachtung des Anlagenverbunds: Das Programmiersystem soll die Steuerungsprogramme für den Anlagenverbund generieren. Das beinhaltet die Generierung von Kommunikationsbefehlen bei mehreren Steuerungen. Betrachtungsbereiche sind somit die Steuerungs- und Feldebene in der Automatisierungspyramide. Die

Kommunikation zu ME- und ERP-Systemen soll in der aktuellen Systemarchitektur nicht integriert, die notwendige Erweiterbarkeit aber vorgesehen werden.

Offene und modulare Systemarchitektur: Um das Programmiersystem einfach an unterschiedliche Randbedingungen anpassen zu können, muss die zu entwickelnde Systemarchitektur eine Flexibilität bzgl. folgender Größen haben:

- Komplexität und Funktionsumfang: Die Komplexität der Aufgabe und der damit verbundene Funktionsumfang im Programmiersystem müssen adaptierbar sein, um an verschiedene Unternehmensrandbedingungen anpassungsfähig zu sein. Das Programmiersystem soll nur den notwendigen Funktionsumfang besitzen. Die Komplexität bzgl. der steuerungstechnischen Konfiguration des Montagesystems, den herstellerspezifischen Programmiersprachen und den spezifischen Schnittstellen für Funktionalitäten von Teilkomponenten des Montagesystems soll für Nutzer des aufgabenorientierten Programmiersystems in jedem Falle abstrahiert werden. Der Nutzer soll sich auf die Prozessbeschreibung beschränken können. In Abhängigkeit der Kenntnisse des Nutzers sind aber auch Mechanismen vorzusehen, mit denen eine weitere Abstraktion der Aufgabe, z. B. die direkte Generierung des Steuerungsprogramms aus einem Produktmodell, ermöglicht werden können. Auf der anderen Seite sollen auch Nutzer die z. B. Wissen bzgl. des Bewegungsverhaltens von Robotern besitzen, diese einbringen können.
- Die Struktur des aufgabenorientierten Programmiersystems muss die unterschiedlichen Randbedingungen bei der Programmierung einer SPS und eines Industrieroboters berücksichtigen.
- Abgebildete Prozesse: Das Programmiersystem muss einfach mit Planungsfunktionalitäten zur Generierung von Steuerungscode für verschiedene Montageprozesse (z. B. Kleben, Schrauben) erweitert werden können.
- Angebundene Softwarewerkzeuge und Simulationssysteme: Die Architektur des Programmiersystems muss Schnittstellen bereitstellen, um Simulationswerkzeuge, wie z. B. zur Robotersimulation oder andere Softwarewerkzeuge, welche schon im Unternehmen zum Einsatz kommen, einbinden und weiterverwenden zu können.
- Verwendete Ressourcen im Montagesystem: Das Programmiersystem und die zugehörigen Informationsmodelle müssen eine Unabhängigkeit bzgl. der verwendeten Ressourcen und den damit einhergehenden Spezifika unterschiedlicher Hersteller bereitstellen.

- Eingabesysteme: Wegen der Variabilität bzgl. der Komplexität der Aufgabe, und den Unterschieden der möglichen Nutzer, muss auch die Schnittstelle zum Eingabesystem und damit dessen Ausprägung flexibel gestaltet sein.

Modellierungstechnische Anforderungen

Damit der Modellierungsaufwand gering gehalten werden kann, sind bestehende Daten im Engineeringprozess zu verwenden. So sind z. B. die für die Konstruktion der Mechanik und Elektrik notwendigen Daten (Fügefolge, Produktdaten und Geometrie etc.), die dort generierten Daten (3D-Layout, Ablaufplan, etc.) sowie die dort verwendeten Datenformate aus den eingesetzten Entwicklungstools als Eingangsgrößen für die Programmierung zu beachten. Die Generierung des Modells eines Montagesystems sollte möglichst automatisch erfolgen. Dafür ist ein entsprechendes Konzept zu entwickeln bzw. auszuwählen.

Aus den durchgeführten Anforderungsanalysen und dem Stand der Wissenschaft und Technik kann außerdem eine grundlegende Struktur der im Rahmen des Programmiersystems relevanten Informationen abgeleitet werden. Um einen industriellen Einsatz zu ermöglichen, sind außerdem bestehende Modellierungsstandards zu evaluieren und im Rahmen des Konzeptes zu berücksichtigen. Folgende Bestandteile sind bei der Modellierung zu unterscheiden:

Montageprodukt und Prozessbeschreibung: Dieser Teilbereich stellt die Eingangsgrößen (Aufgabenmodell) des Programmiersystems dar. Die notwendigen Informationen werden typischerweise in der Produktentwicklung definiert bzw. während der Montageplanung daraus abgeleitet. Dieser Teilbereich umfasst bzgl. des Produktes dessen Geometrie sowie weitere Merkmale, wie z. B. Gewicht, Werkstoff oder Oberflächeneigenschaften. Bzgl. des Prozesses sind dessen Parameter wie bspw. die Schweißgeschwindigkeit oder Toleranzen, sowie die Prozesssequenz und die darin vorgegebenen Zwangsbedingungen bzgl. einer Prozessreihenfolge abzubilden.

Skills: Im zu entwerfenden Programmiersystem sind Skills als Bindeglied zwischen der Montageaufgabenbeschreibung und der Montagesystembeschreibung vorgesehen. Daher werden an diese besondere Anforderungen gestellt. Für die Verwendung der Skills ist es essentiell, dass ein methodisches Vorgehen entwickelt wird, mit dem die in Montagesystemen vorkommen Skills identifiziert werden können. Die Skills sind so zu wählen, dass sie ein Mapping auf die Funktionalitäten realer Komponenten ermöglichen. Dazu muss ein passendes Verhältnis zwischen Detaillierung und Abstraktion gefunden werden.

Montagesystembeschreibung: Dieser Teilbereich umfasst die Beschreibung des Montagesystems und dessen Konfiguration und kann somit dem Umweltmodell bei der aufgabenorientierten Programmierung zugeordnet werden. Typische Teilbereiche sind: die Geometrie der einzelnen Bestandteile, als auch deren Relationen; die Kinematik; das Verhalten von Anlagenbestandteilen wie Steuerungen; die Konfiguration der Anlage bestehend aus I/O-Plan oder Bus- bzw. Netzwerkarchitektur; Umweltgrößen wie Lichtverhältnisse; durch die Anlagenarchitektur vorgegebener Materialfluss; sowie Eigenschaften wie z. B. Genauigkeit oder Geschwindigkeitsbereiche von Teilkomponenten.

Domänenwissen: Dieser Bereich umfasst das notwendige Wissen für die automatische Generierung des Steuerungscodes aus der abstrakten Aufgabenbeschreibung. Das Domänenwissen ist damit je nach Abstraktion der Aufgabe im Programmiersystem oder durch den Nutzer repräsentiert. Unterschiedene Teilbereiche sind Algorithmen bzw. Methoden zur Generierung von Gesamt- und Teillösungen während des Problemlösungsprozesses (z. B. Planung einer kollision- und singularitätsfreien Bahn) und Prozesswissen wie z. B. Schweißgeschwindigkeiten für spezifische Werkstoffe und Nahteigenschaften.

4 Anforderungsanalyse an die aufgabenorientierte Programmierung im industriellen Umfeld

5 Entwicklung eines adaptierbaren aufgabenorientierten Programmiersystems

5.1 Allgemeines

Basierend auf der Zielsetzung und den im vorherigen Kapitel definierten Anforderungen wurden ein Skill-basiertes Informationsmodell sowie ein adaptierbares, aufgabenorientiertes Programmiersystem entwickelt. Im Folgenden wird nach einem kurzen Überblick über das Programmiersystem zunächst das entwickelte Informationsmodell beschrieben, da dies die Grundlage des Programmiersystems darstellt. Die Grundzüge des Konzepts wurden in BACKHAUS & REINHART (2013) vorgestellt.

5.2 Überblick über das Gesamtkonzept

Zu Beginn dieses Kapitels wird zunächst die übergreifende Architektur des adaptierbaren aufgabenorientierten Programmiersystems vorgestellt. Das Programmiersystem wurde als 3-Schichten-Architektur entworfen (vgl. Abbildung 17). Mit einer Schichtenarchitektur wird eine Trennung der Schichten, z. B. in verschiedenen Laufzeitsystemen ermöglicht, wodurch die einzelnen Schichten unkompliziert ausgetauscht werden können (BALZERT 2011, S. 53).

Die oberste Schicht, die *Präsentationsschicht*, repräsentiert die Benutzerschnittstelle, über die der Nutzer des Systems z. B. Aufgaben beschreiben kann. In der darunterliegenden *Logikschicht* sind die eigentlichen Funktionalitäten des Programmiersystems integriert. Die Bestandteile dieser Schicht sind durch Module weiter strukturiert. Die wesentlichen Funktionalitäten eines aufgabenorientierten Programmiersystems, d. h. die Elemente für die Generierung eines Anwendungsprogrammes aus einer abstrakten Beschreibung der Aufgabe, sind in den Modulen „Planungsmodul“ sowie „Postprozessor und Test“ enthalten. Für die beiden Module existieren darüber hinaus Schnittstellen zu Simulationssystemen, die auch extern angebunden werden können. Die vorgeschlagene Architektur sieht verschiedene Simulationssysteme vor, da in Abhängigkeit der betrachteten Montageprozesse unterschiedliche Funktionalitäten notwendig sein können. Das Modul „Planungsmodul“ repräsentiert den eigentlichen Kern des Programmiersystems, in dem die abstrakte Aufgabe in Teilabläufe und schließlich in eine Skill-basierte Ablaufbeschreibung untergliedert wird. Das Modul „Postprozessor und Test“ umfasst die Übersetzung der Skill-basierten Ablaufbeschreibung in

den spezifischen Code für das jeweilige Zielsystem. Anschließend wird dieser Code in einem Test überprüft. Das Modul „Modellgenerator“ ist Teil der Logikschicht, jedoch nicht zwingend mit den beiden anderen Paketen dieser Schicht in einer Softwareanwendung. Seine Aufgabe ist die Generierung des Umweltmodells aus den Ressourcenbeschreibungen. Die *Datenschicht* auf unterster Ebene beinhaltet zwei wesentliche Elemente. Dies ist zum einen das entwickelte Informationsmodell, welches das Aufgaben- und das Umweltmodell einschließt, und zum anderen die Systemdaten, welche z. B. historische Daten, Domänenwissen oder Laufzeitinformationen des Programmiersystems beinhalten. Da diese Systemdaten für die jeweilige Implementierung spezifisch sind, werden sie im Folgenden nicht beschrieben.

Grundsätzlich werden die Benutzer des Programmiersystems in die Bereiche Experte und Nutzer eingeteilt. Experten erweitern das System und dessen Bestandteile, wohingegen Nutzer produktiv mit dem System arbeiten, um Steuerungsprogramme für ein Montagesystem zu erzeugen.

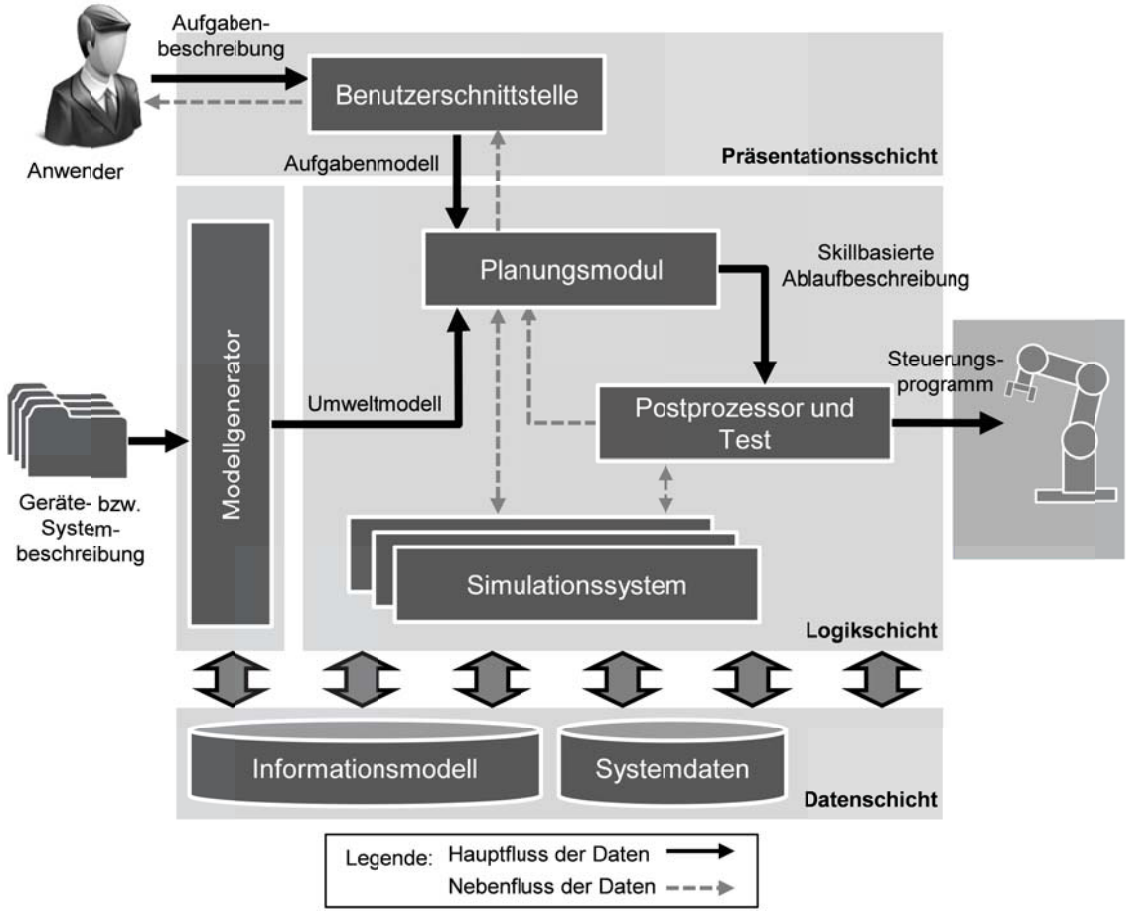


Abbildung 17: 3-Schichten-Architektur des adaptierbaren, aufgabenorientierten Programmiersystems

5.3 Informationsmodell für die aufgabenorientierte Programmierung von Montagesystemen

In der Anforderungsanalyse wurden die Hauptbestandteile eines Informationsmodells herausgearbeitet. Auf Basis dieser Anforderungen wurde ein Konzept sowohl zur generischen als auch detaillierten digitalen Beschreibung der notwendigen Daten für die aufgabenorientierte Programmierung entwickelt, welches in Auszügen in BACKHAUS & REINHART (2015) und BACKHAUS ET AL. (2013) veröffentlicht wurde. In diesem Abschnitt werden zunächst die grundlegenden Bestandteile, Mechanismen und Zusammenhänge im Rahmen des Konzepts hergeleitet, bevor deren Integration in einen bestehenden Beschreibungsstandard dargestellt wird. Als grundlegendes Konzept, um Produktionsanlagen und die dortigen Vorgänge zu beschreiben, wird in einer Vielzahl von Ansätzen das Ressourcen-, Produkte- und Prozesse (PPR)-Konzept genannt (SPUR & KRAUSE 1997; DRATH 2010; HOLLMANN 2013). Dieses Grundprinzip wird auch für den hier beschriebenen Anwendungsfall der aufgabenorientierten Programmierung aufgegriffen und um die Skills erweitert. Das Informationsmodell mit seinen wesentlichen Bestandteilen ist in Abbildung 18 dargestellt. Im Folgenden werden die wesentlichen Bestandteile der Modellierung definiert und abgegrenzt.

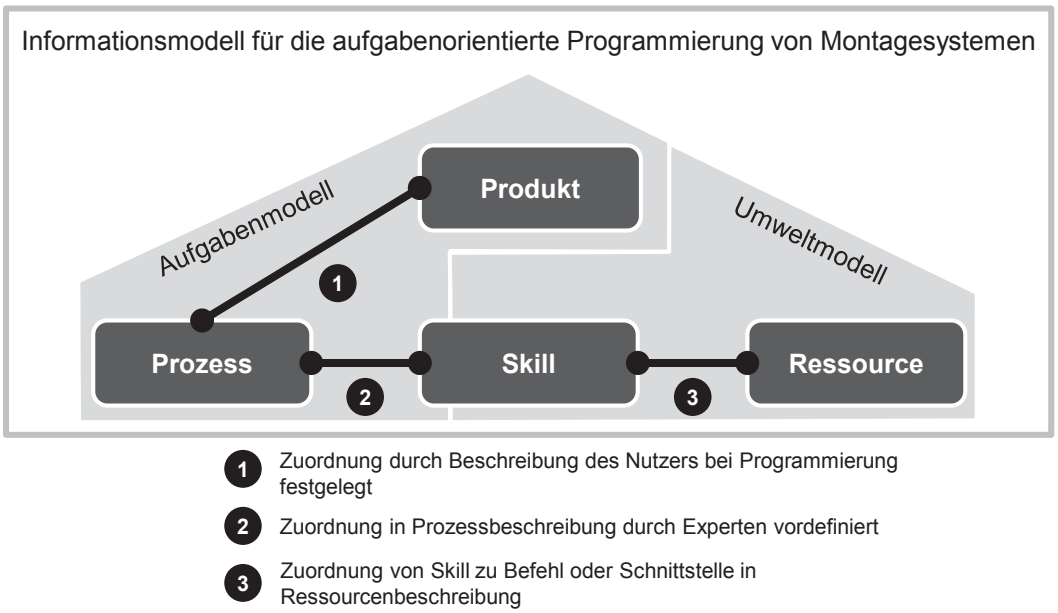


Abbildung 18: Informationsmodell für die aufgabenorientierte Programmierung von Montagesystemen nach BACKHAUS & REINHART (2015)

Produkt: Ein Produkt ist ein zu montierendes Objekt, das typischerweise im Anfangszustand und im Endzustand nach einem Prozess beschrieben wird. Zusammen mit der Prozessbeschreibung enthält die Produktbeschreibung alle zur

Montage notwendigen Informationen. „Produkte können aus mehreren Baugruppen bestehen. Diese können wiederum aus Unterbaugruppen oder Einzelteilen bestehen (...)“ (KLUGE 2011).

Prozess: Ein Prozess ist eine abstrakte, lösungsneutrale Beschreibung einer Operation im Montageprozess. In Prozessen werden ein oder mehrere Produkte durch Ressourcen gefügt, geprüft, justiert und transportiert. Da Produkte jeweils in dem Zustand vor und nach einem Prozess beschrieben werden, bestehen Beziehungen zwischen Prozessen und Produkten. Für die Beschreibung von Fügeverbindungen wie Schweißnähten werden zusätzliche Merkmale verwendet. Prozesse sind lösungsneutral und können daher je nach Art des ausführenden Systems durch eine oder mehrere Ressourcen bzw. deren Skills ausgeführt werden. In dem hier entwickelten Konzept werden in der Spezifikation eines Prozesses auch die notwendigen Skills definiert. Prozesse können auch eine Beschreibung der notwendigen Reihenfolge der zugehörigen Skills enthalten und werden anwendungsspezifisch durch Experten definiert.

Skill: Wie in Kapitel 2 definiert, sind Skills eine semantische und herstellerunabhängige Beschreibung der über Befehle, Funktionen, Funktionsbausteine oder Schnittstellen einer Ressource bereitgestellten Funktionalitäten einer Ressource, welche den Montageprozess unterstützt. Skills sind damit immer Ressourcen zugehörig und aus Sicht einer abstrahierten Ressource beschrieben. Sie umfassen auch weitere Informationen (Eigenschaften) der Ressourcen, die bzgl. der Verwendung der Skills für eine Aufgabe relevant sind. Diese können je nach Skill sowohl statische (z. B. Arbeitsbereiche) als auch dynamische Eigenschaften (z. B. das Verhalten) beschreiben. Mit diesen Inhalten sind eine Überprüfung der Einsatzbarkeit und die Planung der Verwendung im Montageprozess möglich. Um die Verwendung für die Programmierung zu ermöglichen, müssen Skills außerdem auf die Befehle oder Kommunikationsschnittstellen einer der zugehörigen Ressource abgebildet werden. Skills werden in die übergreifenden Klassen Kommunikations-, Verwaltungs- und Applikations-Skills unterschieden, welche aus den Funktionsklassen von SELIG (2011) abgeleitet sind. Die ersten beiden unterstützen nur indirekt den Produktionsprozess bzw. haben Einfluss darauf (indirekte Skills), die letztere unterstützt ihn direkt (direkte Skills). Im Gegensatz zu SELIG (2011) umfassen direkte Skills nicht nur eine Funktion einer Ressource, sondern auch deren beschreibenden Eigenschaften. Im Rahmen dieser Arbeit wurde daher sowohl eine Methode zur Herleitung der direkten Skills, als auch deren Struktur definiert. Verwaltungs-Skills sind meist Applikations-Skills zugeordnet, wie z. B. die Einstellung, ob der Endpunkt der Bewegung relativ oder absolut beschrieben wird. Um ein offenes Konzept zur Zuordnung von Skills zu Ressourcen zu ermöglichen, wurde eine Skill-Hierarchie und eine Grundmenge

an Skills definiert (BACKHAUS ET AL. 2013), deren Herleitung in Abschnitt 5.3.3 beschrieben ist. Skills können sowohl für eine konkrete Ressource instanziiert, als auch als abstrakter Skill in einer Montageprozessbeschreibung verwendet werden. Unter einem instanziierten Skill ist ein Skill zu verstehen, dessen Eigenschaften und Größen für eine konkrete Ressource belegt wurden. Es werden daher übergreifende Skills und instanziierte Skills einer Ressource unterschieden.

Ressource: Ressourcen sind „(…) Maschinen, Anlagen oder deren Komponenten. Sie transportieren, be- oder verarbeiten Teilprodukte zu einem Endprodukt“ (DRATH 2010). Für das Themengebiet der Montage wird diese Definition leicht angepasst: Primäre Ressourcen sind Maschinen, Anlagen oder deren Komponenten, die direkt zum Montageprozess beitragen. Sie können Befehle oder Informationsschnittstellen anbieten, die sie für die Ansteuerung zur Verfügung stellen. Beispiele sind Greifer, Förderbänder oder Industrieroboter. Sekundäre Ressourcen, die nicht direkt zum Montageprozess beitragen, sind z. B. Schutzzäune.

Aufgabenmodell: Das Aufgabenmodell stellt die durch den Nutzer definierte Abbildung der Aufgabe, repräsentiert durch einzelne Produkte, Prozesse oder Skills sowie deren möglichen Sequenzen dar. Zusammen mit dem Aufbau eines Prozesses orientiert sich dieses Konzept also an der VDI/VDE 3682. Mehrere Abstraktionslevel erlauben die Auswahl eines für den Anwendungsfall passenden Abstraktionsgrades.

Umweltmodell: Das Umweltmodell stellt eine Abbildung des gesamten Montagesystems, sowohl repräsentiert durch einzelne Ressourcen und deren Skills als auch deren geometrische und kinematische Beziehungen sowie das räumliche Umfeld, dar. Im Umweltmodell sind darüber hinaus die Umweltbedingungen wie z. B. Lichtverhältnisse, Temperatur oder Luftfeuchtigkeit beschrieben.

In den nachfolgenden Abschnitten werden die Bestandteile des Informationsmodells erläutert. Bevor die Modellierung der Skills beschrieben wird, erfolgt die Beschreibung eines Vorgehens zur Identifikation von direkten Skills. Diese sind direkt an der Umsetzung der Prozesse beteiligt und haben daher besondere Relevanz.

5.3.1 Modellierung der Produkte

Die Beschreibung des Produktes besteht neben der Geometrie und weiteren Merkmalen auch aus dessen Position und Orientierung im Montagesystem. Die Position und die Orientierung des Bauteils müssen nicht zwangsläufig exakt definiert sein. Diese Größen könnten bspw. erst durch ein Kamerasystem bestimmt werden, es müssen aber in jedem Fall Bereiche für die Position definiert sein, da sonst keine Überprüfung möglich wäre, ob das Objekt überhaupt durch ein Kamerasystem detektiert werden kann. Produkte können darüber hinaus hierarchisch in Baugruppen und einzelne Bauteile untergliedert werden. Sowohl Produkte und Baugruppen, als auch Bauteile können Ein- und Ausgangsgrößen eines Prozesses sein. Ein Produkt bzw. dessen Baugruppe oder Bauteile setzen sich aus folgenden Bestandteilen zusammen:

Geometrie: CAD-Geometrie der Baugruppen oder Einzelbauteile.

Merkmale: Zusätzliche produktbeschreibende Merkmale, welche auch in der DIN 4002 und im Merkmalslexikon (MIEHE & MÜLLER 2010) standardisiert sind. Aus der DIN 4002-5 wurden wesentliche qualitative Merkmale extrahiert, welche für den Anwendungsfall der Programmierung relevant sind. U.a. wurden bisher folgende Merkmale definiert: Masse [kg], Massenträgheitsmoment [kg m²], Schwerpunkt im Bezug zur Geometrie, Werkstoff eingeteilt nach entsprechenden Nummernsystemen (z. B. Kunststoffe nach DIN EN ISO 1043, Gusseisen nach DIN EN 1560 oder Aluminium und Aluminiumlegierungen nach DIN EN 573-1), Oberflächenbeschaffenheit bzw. Rauheitswert und Toleranzklasse (z. B. DIN ISO 2768-1).

Global Frame: Dieser wird beschrieben durch die Endposition x, y, z Koordinaten [mm] im globalen Koordinatensystem, sowie die Orientierung [°] nach z, x‘, z‘‘-Konvention. Die Position kann darüber hinaus auch als Wertebereich angegeben werden, wenn z. B. die genaue Position auf einem Tisch nicht bekannt ist.

Flag Position: Dieser gibt an, ob der Global Frame exakt bekannt ist. Falls dies nicht der Fall sein sollte, wird zusätzlich ein Wertebereich für den Global Frame, wie z. B. eine Fläche angegeben. Je nach Genauigkeit des Global Frames können z. B. auch Justagevorgänge notwendig sein.

5.3.2 Modellierung der Prozesse

Prozesse sind lösungsneutrale, d. h. unabhängig vom Montagesystem, Beschreibungen einer Montageaufgabe in der auch die notwendigen Skills definiert sind. Da ein einzelner Prozess ein Element der Aufgabenbeschreibung darstellt, orientieren sich dessen Bestandteile an den Teilmengen eines Aufgabenmodells, wie es z. B. bei MUCKENHIRN (2005), MUNZERT (2010) oder HATWIG (2014) Verwendung findet. Ein Prozess setzt sich aus folgenden Bestandteilen zusammen:

Prozess-Anteil: In diesem Teil können Größen spezifiziert werden, die nicht in der Produktbeschreibung für Anfangs- und Endzustand enthalten sind. Dieser Bestandteil des Aufgabenmodells unterscheidet sich je nach Prozess. Teilweise müssen in diesem Bereich keine weiteren Angaben gemacht werden. Ein Beispiel ist das Einlegen (nach DIN 8593-1). Für diesen Prozess ist nur die Handhabung der Bauteile notwendig. Beispiele für Prozessparameter die angegeben werden, sind z. B. Drehmomente beim Verschrauben oder die freie Drahtlänge beim MIG/MAG Schweißen.

Geometrie-Anteil: Anfangs- und Endzustand der Produkte vor und nach dem Prozess beschreiben nicht zwangsläufig die Geometrie der Fügeverbindung. Diese geometrischen Beschreibungen sind spezifisch für die jeweiligen Prozesse, wie z. B. die Nahtart oder die Nahtdicke einer Schweißnaht. Zur Beschreibung werden Standardelemente wie Volumenmodelle, Frames, Flächen, Geraden, Kreise und Splines verwendet. Der Geometrie-Anteil ist als Ausgangsgröße des Prozesses definiert und muss in Relation (Position und Orientierung) zur Geometrie des Produktes bzw. der Produkte stehen.

Zugehörige Skills: Die für einen Prozess benötigten Skills werden an dieser Stelle hinterlegt. Dabei existieren zwei Möglichkeiten der Beschreibung. Entweder als einfach Liste der Skills oder zusätzlich mit deren notwendigen Sequenz für den Prozess.

Abbildung 19 zeigt den Prozess „Schrauben“. Vor der Verwendung im Programmiersystem müssen die Prozesse durch Experten definiert werden. Dem Nutzer des Programmiersystems stehen die Prozesse in einer Bibliothek zur Verfügung. Für jeden Prozess sind zugehörige Planungsfunktionalitäten im aufgabenorientierten Programmiersystem notwendig. Bspw. um einen Handhabungsvorgang zu planen. Im Rahmen dieser Arbeit wurden eine Reihe wichtiger Basis-Prozesse definiert. Dies sind „Zusammensetzen“, „Schweißen mit elektrischem Strom“, „Handhaben“ und „Position messen“.

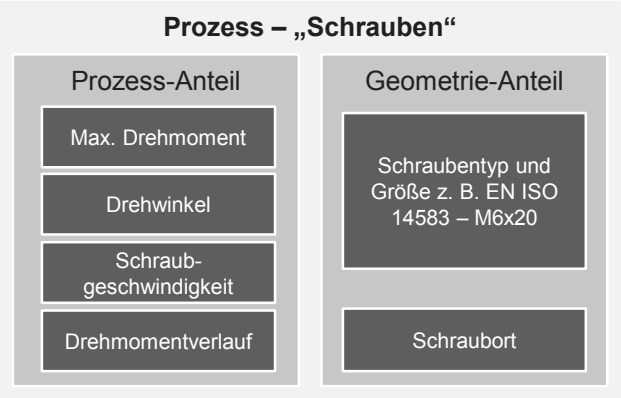


Abbildung 19: Prozess- und Geometrie-Anteil des Prozesses „Schrauben“

Prozesse können durch den Experten durch sechs wesentliche Schritte definiert werden, die nachfolgend erläutert und in Abbildung 20 dargestellt sind. Im ersten Schritt wird durch den Experten der grundsätzliche Prozess festgelegt und beschrieben. Dies erfolgt anhand der Beschreibungen, die in Normen wie DIN 8593 bzw. VDI-Richtlinie 2860 existieren. Im zweiten Schritt werden die für den Prozess notwendigen Skills bestimmt bzw. ausgewählt.

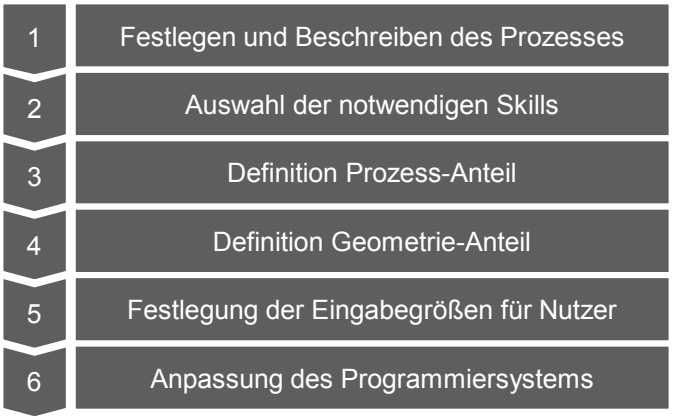


Abbildung 20: Vorgehen zur Definition von Prozessen

Dabei besteht auch die Möglichkeit auf abstraktere Skill-Klassen (z. B. „Bewegen auf geführter Bahn“ statt „Bewegen auf geführter linearer Bahn“) zu verweisen, um den Lösungsraum an Ressourcen nicht einzuschränken. Mit dieser Zuordnung der Skills können indirekt auch Anforderungen an die später verwendeten Ressourcen abgebildet werden. Für die Zuordnung sind das Prozesswissen des Experten und das Klassendiagramm der Skills notwendig. Darüber hinaus ist es auch möglich, eine dem Prozess zugehörige Reihenfolge an Skills festzulegen. Für viele Prozesse kann auch die notwendige Sequenz der Skills vorab definiert werden. Beispiele für wiederkehrende prozessspezifische Abläufe für die Monta-

ge finden sich z. B. bei WIELAND (1995). Ein Beispiel ist in Abbildung 21 dargestellt. Im dritten und vierten Schritt werden Prozess- und Geometrieanteil des Prozesses identifiziert und festgelegt. Für den Prozess Kleben sind dies z. B. eine Spline Bahn für den Geometrie Anteil und der Kleberaupendurchmesser. Für den Prozessanteil sind zunächst die Parameter zu sammeln und anschließend ist festzulegen, welche Größen vom Nutzer vorgegeben werden und welche durch das Programmiersystem automatisch generiert werden. Für den Geometrieanteil sind geeignete Geometrieelemente wie Punkte, Frames, Flächen, Geraden oder Volumenelemente auszuwählen. Im letzten Schritt werden die vom späteren Nutzer bei der Verwendung des Prozesses benutzten Eingangsgrößen festgelegt, die sich auf den Prozess und den Geometrieanteil beziehen. Damit wird festgelegt, auf welchem Abstraktionsniveau die Programmierung erfolgt und welches Wissen der Nutzer vorweisen muss. Im letzten Schritt muss das Programmiersystem mit den notwendigen Funktionalitäten für den Prozess ausgestattet werden. Es ist essentiell, die Prozessbeschreibungen nicht nur singulär für die aufgabenorientierte Programmierung zu verwenden, sondern im gesamten Engineering bzw. Montageplanungsprozess zu verankern. Dies ermöglichte den direkten Import der Prozessbeschreibungen aus frühen Planungsphasen in das aufgabenorientierte Programmiersystem.

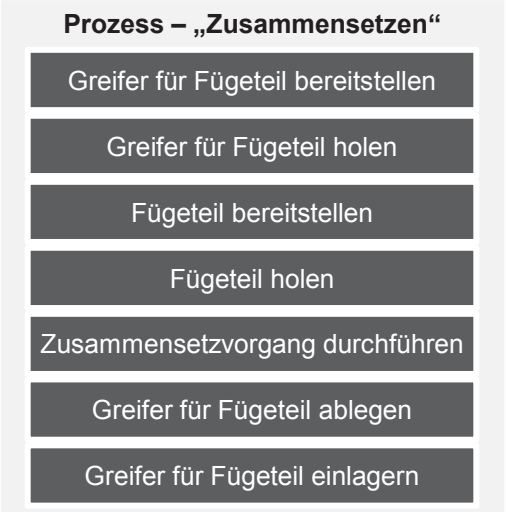


Abbildung 21: Ablauf des Prozesses „Zusammensetzen“ in Anlehnung an WIELAND (1995)

5.3.3 Vorgehen zur Identifikation von direkten Skills

5.3.3.1 Erläuterung der Problemstellung und Lösungsansatz

Zum besseren Verständnis der Problemstellung, ist diese in Abbildung 22 dargestellt. Auf abstrakter Ebene bieten Ressourcen Befehle oder Kommunikationsschnittstellen an, über deren Ein- und Ausgangsgrößen die dahinterstehende Funktionalität in Hard- oder Software angesprochen werden kann. Diese Funktionalität, inklusive deren Ein- und Ausgangsgrößen sowie Eigenschaften, wird als Skills bezeichnet und soll unterschieden werden.

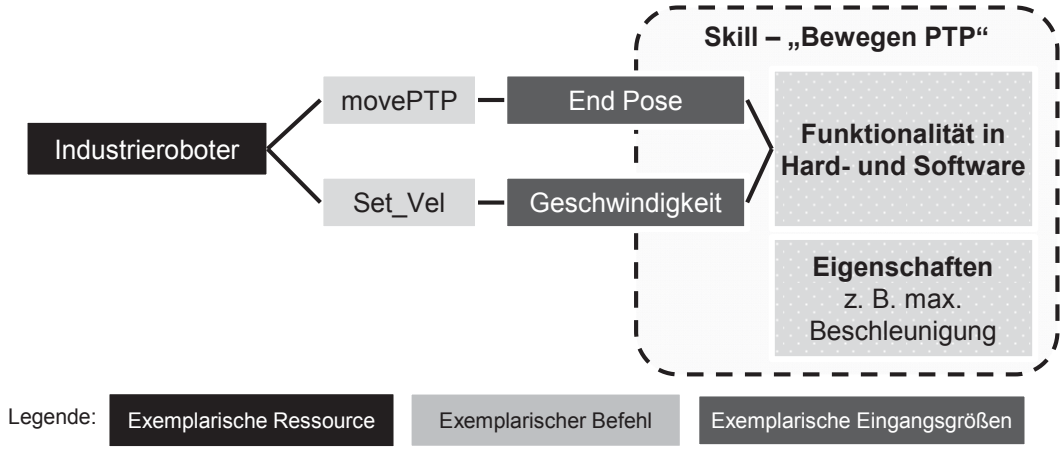


Abbildung 22: Verschaulichung der Problemstellung der Modellierung von Skills an einem Beispiel nach BACKHAUS & REINHART (2015)

Zum Clustering[2], zur Typisierung oder zur Mustererkennung werden Merkmale verwendet, die einzelne Elemente trennen, bzw. eine Zuordnung zu Klassen möglich machen, sogenannte trennende Merkmale (DUDA ET AL. 2001). Es müssen also Merkmale identifiziert werden, an denen die Funktionalitäten bzw. Skills getrennt werden können, um Klassen von Skills zu bilden. Die Identifikation trennender Merkmale ist meist ein empirisches Vorgehen (DUDA ET AL. 2001). Aus Sicht der Ressourcen lässt sich eine große Anzahl an Merkmalen identifizieren, die aber nicht zwangsläufig als trennendes Merkmal Verwendung finden sollten, wie z. B. die Traglast eines Industrieroboters. Aufgrund der Heterogenität von Industrierobotern wäre eine zu feine Trennung der Klassen die Folge. Es

[2] Da in der Literatur widersprüchliche Definitionen existieren, werden für diese Arbeit die folgenden Begriffsbestimmungen verwendet: Unter Clustering wird die Einteilung einer Menge von Objekten in einer Gruppe gleichartiger Objekte verstanden (ESTER & SANDER 2000). Im Gegensatz dazu ist unter der Klassifikation die Einteilung von Objekten in schon vorhandene Klassen zu verstehen (BACKHAUS ET AL. 2011).

muss daher ein Ansatz gefunden werden, wie trennende Merkmale identifiziert werden können.

Da die Ressourcen und ihre Funktionalitäten im Montageprozess zum Einsatz kommen, besteht die Möglichkeit die grundsätzlichen Merkmale und damit die Funktionalitäten aus den Montageprozessen abzuleiten. In den Montageprozessen bzw. -funktionen lassen sich Zustände, Zustandsübergänge und Zustandserfassungen mit physikalischen und logischen Größen klar voneinander abgrenzen (SCHMIDT 1992). Diese können den Funktionalitäten zugeordnet werden, müssen jedoch aus der Sicht einer abstrahierten Ressource betrachtet werden.

Im Folgenden wird das Vorgehen zu einer überschneidungsfreien Grundhierarchie an Skills beschrieben. In den ersten drei Schritten werden die Skills ohne Betrachtung von realen Ressourcen unterschieden. Daher werden auch nur zwingende Ein- und Ausgangsgrößen betrachtet. Erst in den weiteren Schritten werden reale Ressourcen bzw. deren Schnittstellen und Befehle mit den zugehörigen Ein- und Ausgangsgrößen betrachtet. Abbildung 23 zeigt das beschriebene Vorgehen. Dieses ist zwar zu Teilen automatisierbar, die Identifikation der trennenden Merkmale ist aber ein manueller Prozess.

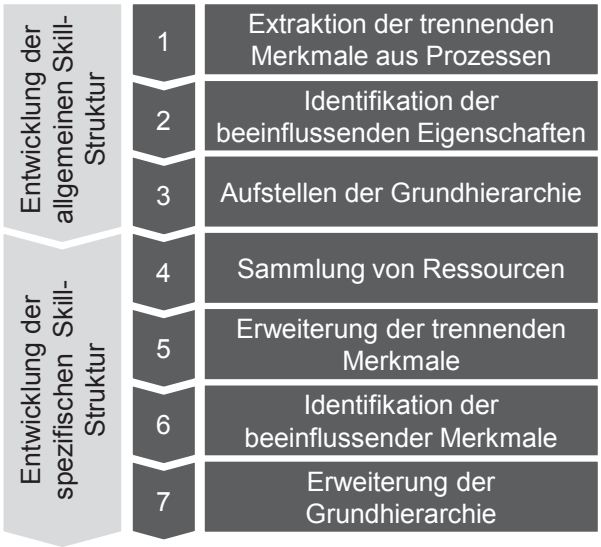


Abbildung 23: Vorgehen zur Identifikation von direkten Skills und deren Struktur nach BACKHAUS & REINHART (2015)

5.3.3.2 Extraktion trennender Merkmale aus Prozessen

Die Extraktion der trennenden Merkmale erfolgt auf mehreren Ebenen. Die in Normen u. a. VDI-RICHTLINIE 2860, DIN 8593 beschrieben Fertigungsverfahren und Funktionen bilden die Grundlage. Allerdings beschreiben diese Prozesse

bzw. Funktionen Zustände, Zustandsänderungen und Zustandserfassungen aus Sicht von Produkten. Da diese außerdem nicht auflösbare Überschneidungen aufweisen (SCHMIDT 1992), können sie nicht direkt übernommen werden. Beispielsweise kann das Fertigungsverfahren „Zusammensetzten“ aus der DIN 8593-1 auch durch die Einzelfunktionen „Bewegen“ und „Sichern“ der VDI-RICHTLINIE 2860 dargestellt werden. SCHMIDT (1992) leitet für die Planung von Montagesystemen fünf grundlegende Zustände, Zustandsänderungen und Zustandserfassungen ab: *Speichern, Bewegen, Verbinden, Verändern* und *Vergleichen* und er benennt für diese auch die wesentlichen physikalischen Einflussgrößen. Dieser Grundgedanke wird aufgegriffen und für den Anwendungsfall angepasst und detailliert. Auf oberster Ebene werden die folgenden fünf Klassen an Skills unterschieden, als trennendes Merkmal wird die Bereitstellung der grundlegenden Zustände, Zustandsänderungen und Zustandserfassungen herangezogen (vgl. BACKHAUS & REINHART (2015)):

Speichern: Speichern ist das zur Verfügung stellen der Funktionalität für das temporäre Aufrechterhalten einer unterschiedlich exakt definierten Position und Orientierung eines Objekts im Bezug zu einem Referenzkoordinatensystem. Bestimmend ist hier die Aufrechterhaltung eines Orientierungszustands nach VDI 2860 durch Kraft- und Momentengleichgewichte (SCHMIDT 1992). Wie diese Gleichgewichte aufrechterhalten werden, wird auf dieser Ebene zunächst nicht betrachtet.

Bewegen: Im Gegensatz zur Definition von SCHMIDT (1992) und der VDI-RICHTLINIE 2860 wird eine Bewegung als eine Änderung der Position und Orientierung eines Teils oder mehrerer Teile einer Ressource in Bezug zu einem Referenzkoordinatensystem verstanden. Für die Bewegung gelten die Gesetze der Dynamik und Kinematik (SCHMIDT 1992). Die bewegten Teile der Ressource haben typischerweise eine kinematische Verbindung zu einer Ressource oder einem Teil einer Ressource, welche eine der vier anderen Klassen an Skills zur Verfügung stellt, wie z. B. die Verbindung eines Industrieroboters zu einem Greifer. Eine weitere Möglichkeit ist das Zurverfügungstellen von Kontaktflächen für Produkte über Schwerkraft, wie z. B. bei einem Förderband.

Verbinden: Bereitstellen der Funktionen zum Herstellen einer Verbindung zwischen zwei Verbindungspartnern. Durch die Verbindung wird ein Kräftegleichgewicht in der Verbindungsstelle durch Stoff-, Form- oder Kraftschluss bzw. stoffschließende, energetische oder chemische Maßnahmen erzeugt (SCHMIDT 1992). Nur die notwendigen Bewegungen von Hilfselementen wie Schrauben, sind dieser Zustandsänderung zugeordnet. Die für den Prozess notwendigen Bewegungen der Fügepartner zueinander und die Bewegung der Ressource, um

die Zustandsänderung „Verbinden“ an verschiedenen Positionen durchzuführen, sind nicht Bestandteil. Somit ist das „Zusammensetzen“ zweier Bauteile aus der DIN 8593-1 nicht dem „Verbinden“ zuzuordnen, sondern wird durch „Bewegen“ und „Speichern“ dargestellt.

Verändern: Diese Klasse umfasst diejenigen Funktionalitäten, welche durch eine Zustandsänderung zwar keine Verbindung herstellen, jedoch den Zustand eines Objektes anderweitig verändern können (SCHMIDT 1992). Für die Montage sind das im Wesentlichen die dem Bereich Sonderoperationen, wie z. B. dem Erhitzen, zugeordneten Änderungen.

Vergleichen: Vergleichen umfasst das Bereitstellen aller Zustandserfassungen, die durch das Erfassen, das Gegenüberstellen mit einer Referenz und das Initiieren eines Rückgabewertes weiter unterteilt werden können. Erfasst werden physikalische (Kraft, Position, etc.) oder chemische Zustände bzw. Zustandsänderungen (SCHMIDT 1992).

Diese fünf Meta-Klassen der Skills können durch weitere trennende Merkmale detailliert werden. Die trennenden Merkmale sind jedoch für jede einzelne Meta-Klasse spezifisch. Auf Basis bestehender Normen sowie weiteren physikalischen Größen wurden alle übergreifenden Klassen in mehreren Ebenen detailliert und unterschieden (vgl. BACKHAUS & REINHART (2015)). Die physikalischen oder chemischen Unterscheidungen wie z. B. Energieträger beim Löten, sind damit weitere trennende Merkmale für die abgeleiteten Klassen. Für die Unterteilung können jedoch nicht immer direkt die Normen herangezogen werden. An dieser Stelle wird die Unterteilung nur an einem Beispiel gezeigt, die weiteren trennenden Merkmale sind im Anhang A2 zu finden, das Ergebnis, d. h. die übergreifenden Skill-Klassen, wird in Schritt 3 des Vorgehens beschrieben. Die Zustandsänderung „Bewegen“ wird zunächst nach der Veränderbarkeit der Bahn der kinematischen Schnittstelle oder Kontaktfläche unterschieden (vgl. Abbildung 24).

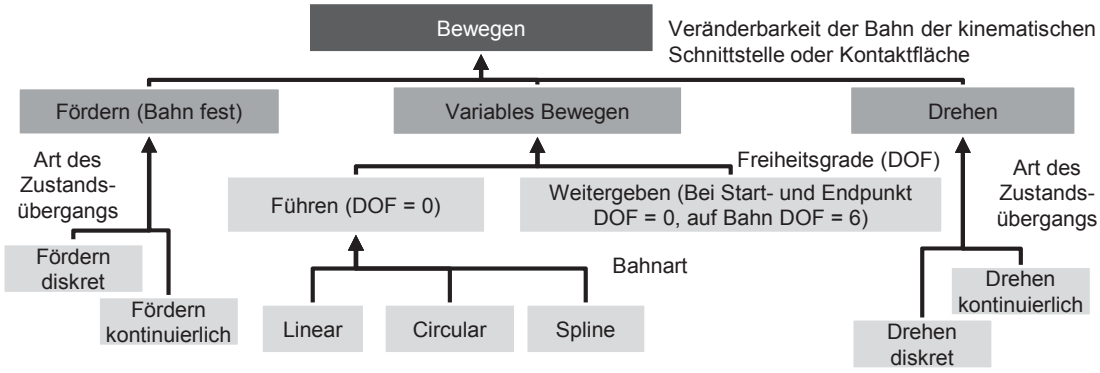


Abbildung 24: Unterteilung des Skills „Bewegen“ mit trennenden Merkmalen nach BACKHAUS & REINHART (2015)

Geführte Bewegungen können des Weiteren nach der Art der Bahn unterschieden werden. Diese Trennung orientiert sich weitestgehend an der VDI-Richtlinie 2860. Für die Oberklasse „Verbinden“ kann teilweise eine Trennung nach den in der DIN 8593 definierten Unterteilungen erfolgen.

5.3.3.3 Identifikation der beeinflussenden Eigenschaften

Um planen zu können, ob und wie der Skill einer Ressource für einen spezifischen Prozess verwendet werden kann, müssen alle Größen der Ressource bekannt sein, die Einfluss auf die Funktionalität haben. Diese können zwischen Ein- und Ausgangsgrößen (z. B. Zielposition einer Bewegung) und feststehenden Größen (z. B. Einfluss der Kinematik auf Bewegung) unterschieden werden. Diese sind zumeist weitere physikalische, chemische oder logische Größen. Nur ein Teil der Eingangsgrößen ist zwingend notwendig (z. B. Endpose einer Bewegung), andere Teile sind entweder eine Eingangsgröße oder eine feststehende Größe (z. B. Beschleunigungsprofil einer Bewegung) in Abhängigkeit von der Ressource. Um diese Eigenschaften aufzudecken, ist keine direkte Ableitung aus den Prozessen möglich. Sie müssen aus der Betrachtung der einzelnen Skill-Klassen identifiziert werden. Für alle Größen werden auch Genauigkeit und Einheit festgelegt. Die Zuordnung erfolgt nicht auf Ebene der Meta-Klassen, sondern wird erst auf den darunter liegenden Ebenen angewendet. Um das Vorgehen zu unterstützen, wurde ein Katalog an Größen aufgestellt, der sich in folgende Bereiche unterteilen lässt:

- Identifikation der Größen aus dem Bereich der klassischen Mechanik (Längen bzw. damit beschriebene Geometrien, Masse, Zeit, Geschwindigkeit, Beschleunigung, …)
- Identifikation der Größen aus dem Bereich Elektrodynamik (Stromstärke, Spannung, elektrische Feldstärke, magnetische Feldstärke, Optische Kenngrößen, …)
- Identifikation der Größen aus dem Bereich Thermodynamik (Temperatur, Druck, Massenstrom, …)
- Identifikation der informationstechnischen Größen (Rechenoperationen pro Zeit, …)

Beispielhafte Eigenschaften bzw. beeinflussende Merkmale für ein Kamerasystem sind die Schockresistenz, das Farbspektrum, die Brennweite, die Auflösung und die Framerate. Da in diesem Schritt noch keine konkrete Ressource betrach-

tet wird, kann nicht festgelegt werden, ob diese Größen veränderbar oder feststehend sind.

5.3.3.4 Aufstellen einer Grundhierarchie

Aus den ersten beiden Schritten leitet sich eine Klassen-Hierarchie der grundlegenden Skills in Montagesystemen mit deren wesentlichen Eigenschaften ab. Abbildung 25 zeigt einen Ausschnitt der Klassen. Dies beinhaltet die Vererbung von Ein- und Ausgängen sowie Eigenschaften auf die jeweils abgeleiteten Klassen. Unter der Vererbung ist die Übernahme der Attribute und Eigenschaften von der höheren Klasse in die niedrigere zu verstehen (RUMPE 2004, S. 23). Eine detaillierte Aufstellung der Klassen mit trennenden Merkmalen ist im Anhang A2 zu finden.

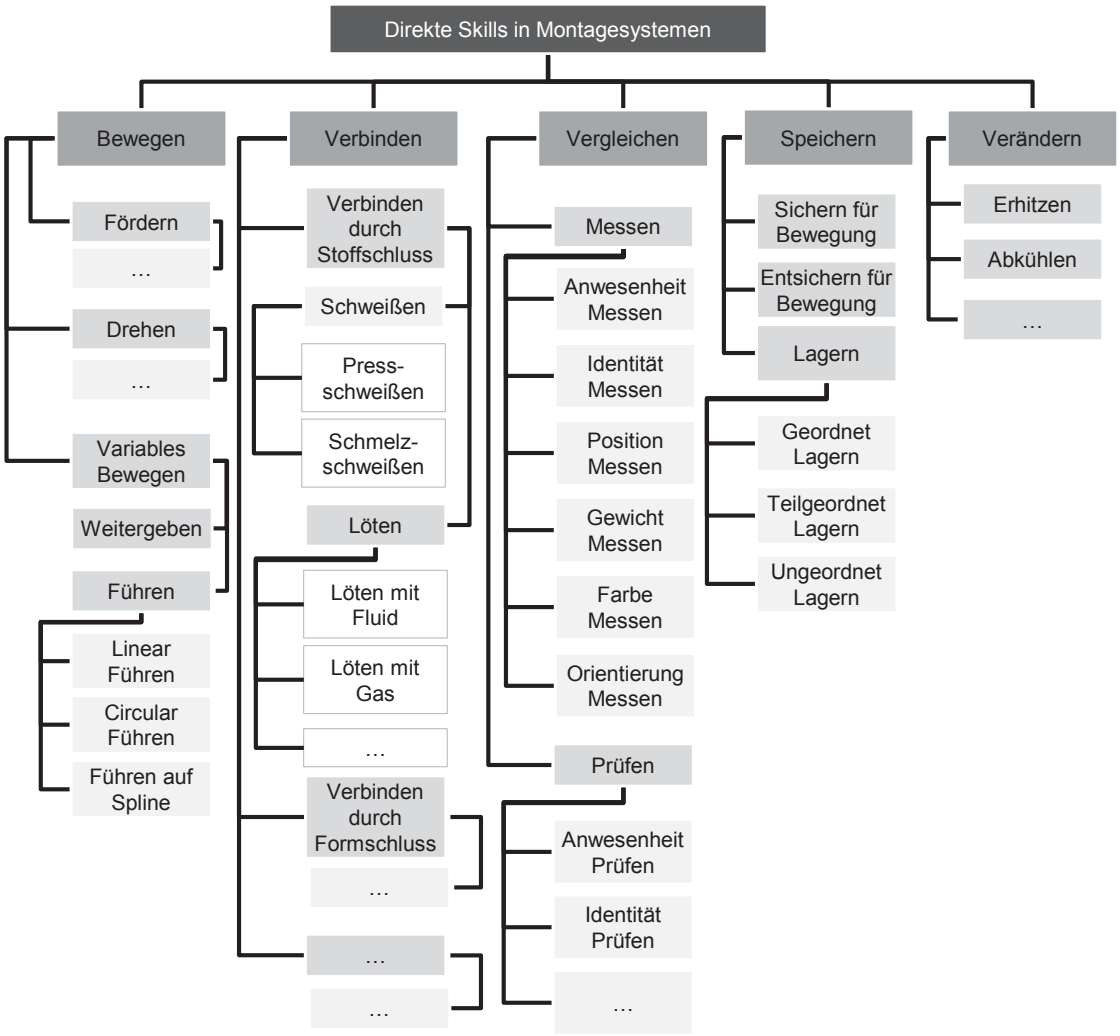


Abbildung 25: Auszug der Klassen an Skills in Montagesystemen nach BACKHAUS & REINHART (2015)

5.3.3.5 Sammlung von Ressourcen

Um die allgemeinen Skills weiter detaillieren zu können und auch insbesondere die Ein- und Ausgangsgrößen realer Ressource zu berücksichtigen, erfolgt in diesem Schritt zunächst eine Sammlung von Ressourcen. Die Ein- und Ausgangsgrößen haben so große Relevanz, weil diese vom Programmiersystem erzeugt bzw. verwendet werden müssen. Das Vorgehen wurde im Rahmen dieser Arbeit nur für einen Ausschnitt an möglichen Ressourcen durchgeführt. Es bietet jedoch den Rahmen, um die Hierarchie umfassend aufzubauen. Grundsätzlich können in diesem Schritt verschiedenste Ressourcen ausgewählt und betrachtet werden, ohne vorher die Zuordnung zu einem Prozess zu betrachten, da die Zuordnung anhand der Grundhierarchie erfolgt. Im Rahmen dieser Arbeit wurden folgende Ressourcentypen von jeweils unterschiedlichen Herstellern betrachtet: Industrieroboter, Greifer, Förderbänder, Lichtschranken, Schweißbrenner inkl. Steuerung, Dreh-Kipp-Tisch als Positioniersystem, Spannsysteme, Kamerasystem mit Programm zur 3D-Positionserkennung. Diese bilden die zu vergleichenden Elemente bei der Entwicklung der Hierarchie.

5.3.3.6 Erweiterung der trennenden Merkmale

Die Ein- und Ausgangsgrößen einer Gruppe an Ressourcen, die den gleichen Skill besitzen, werden identifiziert und gesammelt dem jeweiligen Skill zugeordnet. Diese umfasst also alle Ein- und Ausgangsgrößen der betrachteten Ressourcen, womit garantiert ist, dass bei einer Planung mit diesem Skills auch alle Ressourcen komplett abgedeckt werden. Es erfolgt dabei jedoch eine Unterscheidung. Ein- und Ausgangsgrößen, die bei allen Ressourcen vorkommen, werden als Muss-Merkmale definiert. Alle nur bei einzelnen Ressourcen vorkommenden werden als Kann-Merkmale bezeichnet. Muss-Merkmale sind typischerweise auch in der Prozessbeschreibung schon direkt definiert und müssen in jedem Fall durch das Programmiersystem abgedeckt werden. Mit der Unterscheidung kann der notwendige Funktionsumfang des Programmiersystems reduziert werden. Dies führt jedoch teilweise zu nicht ausgenutzten Einstellungsmöglichkeiten der Ressourcen. Abbildung 26 zeigt ein Beispiel für Ein- und Ausgangsgrößen für zwei Schrauber mit Steuerung sowie den daraus abgeleitete Skill ohne Darstellung der sonstigen Eigenschaften. Da in der übergreifenden Hierarchie Eingangsgrößen und Eigenschaften nicht immer klar zugeordnet werden können, werden diese somit auch festgelegt.

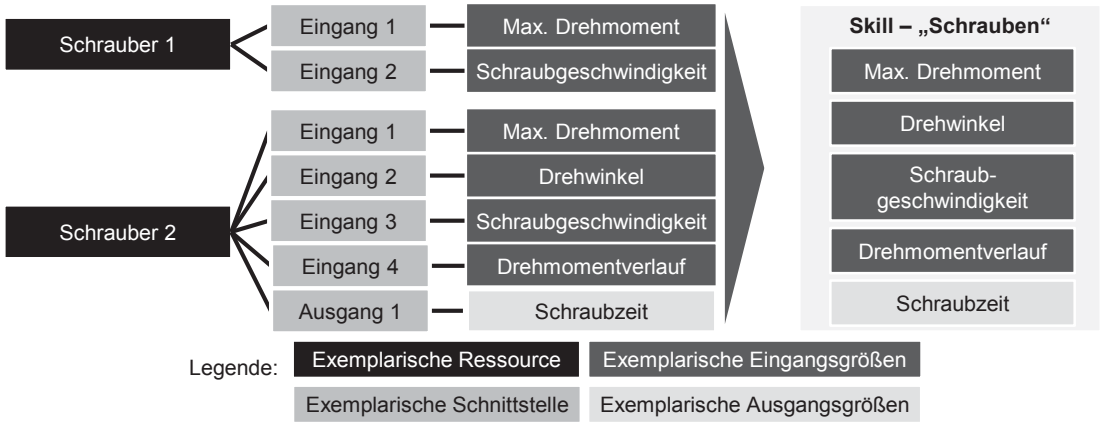


Abbildung 26: Ableitung der trennenden Merkmale aus realen Ressourcen am Beispiel „Schrauben“

5.3.3.7 Identifikation der beeinflussenden Merkmale

Für die konkrete Auswahl einer Ressource sind oftmals weitere Eigenschaften relevant, die aus der Entwicklung der Skills aus den Zuständen, Zustandsänderungen und Zustandserfassungen nicht direkt abgeleitet werden können. Ein Beispiel ist der Linsendurchmesser einer Roboterschweißzange. Diese Größen ergeben sich aus der technischen Umsetzung der Ressourcen. Sie werden daher für jede Ressource erfasst und den zugehörigen Skills gesammelt als Eigenschaften zugeordnet.

5.3.3.8 Erweiterung der Grundhierarchie

Mit der Einordnung der neuen trennenden und beeinflussenden Merkmale in die ressourcenunabhängige Grundhierarchie wird diese weiter spezialisiert. Sollte aus der Betrachtung weiterer Ressourcen eine Veränderung in den Inhalten der Skills notwendig sein, so können diese Skills als Spezialisierung der vorherigen Klassen in die Hierarchie eingeordnet werden.

5.3.4 Modellierung der Skills

5.3.4.1 Direkte Skills

Bei der Modellierung der Skills werden drei Hierarchieebenen unterschieden (vgl. Abbildung 27), welche jeweils eine Detaillierung der oberen Ebene darstellen. Die zwei oberen Ebenen ergeben sich aus dem vorgestellten Vorgehen. Sie beschreiben die Skills auf abstrakter Ebene ohne Betrachtung von Ressourcen bzw. als spezifische Skills nach der Betrachtung der Charakteristika von Ressourcen. Werden Skills für eine spezifische Ressource instanziiert, d. h. mit deren Werten belegt und auf deren Befehle bzw. Kommunikationsschnittstellen abgebildet, ergibt sich eine dritte Hierarchieebene. Die Modellierungsbestandteile auf den oberen beiden Hierarchieebenen unterscheiden sich nicht. Im Folgenden werden die Modellierungsbestandteile der Ebenen vorgestellt.

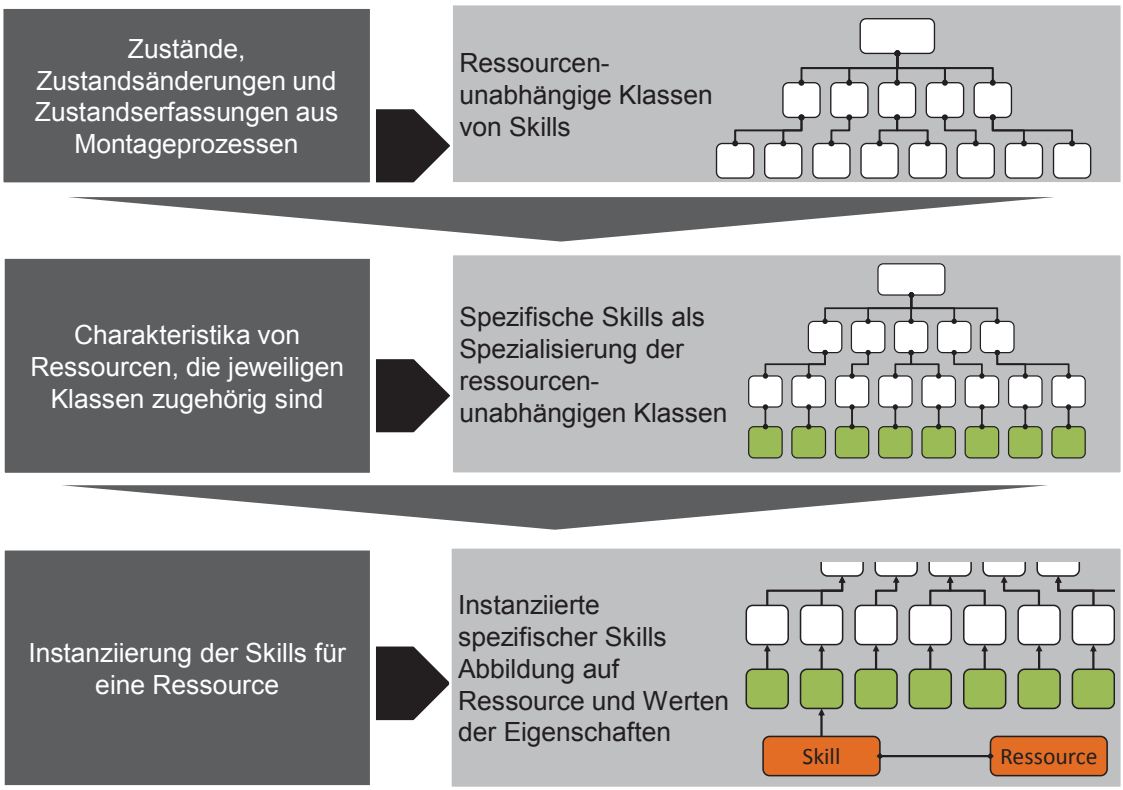


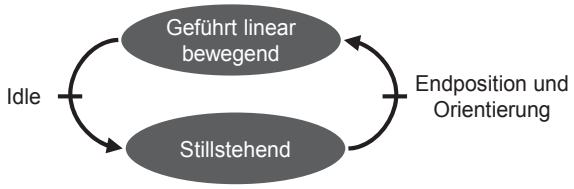
Abbildung 27: Hierarchieebenen von Skills

Parallel zu den Schritten eins bis drei des Vorgehens wurde in dessen ersten Anwendung auch die Grundstruktur eines ressourcenunabhängigen bzw. spezifischen Skills abgeleitet. Da Skills eine Abstraktion der über Befehle und Schnittstellen verwendbaren Funktionalitäten einer Ressource darstellen, liegt es nahe,

ein an die Beschreibung von Funktionen in der Informatik angelehntes Konzept zu verwenden. Dort werden Funktionen meist durch *allgemeine Informationen* (z. B. Name und Beschreibung der Funktion), deren *Eingangsgrößen* (diese verändern die Ausführung der Funktion), deren *Eigenschaften* (feststehende Charakteristika einer Funktion wie maximale Greifkraft) sowie deren *Ausgangsgrößen* beschrieben. Ein Skill lehnt sich somit an dem EVA-(Eingabe, Verarbeitung, Ausgabe)-Prinzip aus der Datenverarbeitung an (HAUN 2007, S. 188). Für alle Bestandteile dieser allgemeinen Skills gilt, dass jeweils Name, Wert, Einheit, Min/Max, Beschreibung und Defaultwert angegeben werden, wenn es sich nicht um kinematische oder geometrische Größen handelt. Als Eingangsgrößen werden nur solche definiert, die auch zwingend für die Ausführung des Skills vorgegeben werden müssen. Zusätzlich zu den allgemeinen Informationen, Ein-, Ausgangsgrößen sowie Eigenschaften wird jeder Skill auch mit einem *Zustandsdiagramm* beschrieben, mit dessen Transitionen auch die Ein- und Ausgangsgrößen verknüpft werden. Diese sind also notwendige Vorbedingungen, um den Skill auszuführen. Tabelle 2 zeigt die Struktur am Beispiel einer „geführten linearen Bewegung“.

Tabelle 2: Bestandteile eines übergreifenden Skills am Beispiel "geführte lineare Bewegung"

Bestandteile	**Geführte lineare Bewegung**
Allgemeine Informationen	Beschreibung: Bewegung entlang einer linearen Bahn. Die Orientierung auf der Bahn ist zu jedem Zeitpunkt bekannt.
Eingangsgrößen	Endposition (x, y, z) [mm], Orientierung (z, x‘, z‘‘) [°]
Ausgangsgrößen	-
Eigenschaften	Kinematik und Geometrie sind in der Ressource beschrieben
	Geschwindigkeit [m/s]
	Beschleunigung [m/s²]
	Beschleunigungsverlauf
	maximal aufnehmbare Kräfte [N]
	maximal aufnehmbares Moment [Nm]

	Traglast [kg]
	Absolutgenauigkeit der Endposition +/-[mm]
	Wiederholgenauigkeit der Endposition +/-[mm]
Zustandsdiagramm	

Bei der Betrachtung des Zustandsdiagramms aus Tabelle 2 ist erkennbar, dass der entsprechende Skill für die Rückkehr in den Zustand „Stillstehend“ keinen Trigger benötigt. Die zugehörige Ressource vollzieht also den Zustandsübergang automatisch nach Ende der Bewegung. Es existieren jedoch auch einige Skills die einen weiteren Eingabewert erwarten, um wieder in den Ausgangszustand zurückzukommen. Ein Beispiel ist das Anhalten eines kontinuierlich laufenden Förderbands.

Ein instanziierter Skill beinhaltet neben den oben genannten Bestandteilen *allgemeine Informationen, Eingangsgrößen, Ausgangsgrößen und Eigenschaften* zusätzlich ein Mapping bzw. eine Abbildung der Ein- und Ausgangsgrößen des Skills auf die Befehle, Schnittstellen oder Funktionsbausteine der jeweiligen Ressourcen. Das *Mapping erfolgt durch sogenannte Pointer*, die unterschiedliche Ausprägung haben können, wie in Abbildung 28 gezeigt ist. Dabei können auch Umrechnungsfaktoren im Vergleich zu in der Hierarchie definierten Einheiten (z. B. Meter auf Millimeter) berücksichtigt werden.

Mit der Abbildung auf die Schnittstellen der Ressource ist somit auch eine Verknüpfung mit deren Verhaltensmodell gegeben. Beim Vergleich der Arten von Pointern für verschiedene Skills ist auffällig, dass zumeist ein Befehl für die Auslösung der Funktionalität verantwortlich ist und mögliche weitere Befehle zwar davor, aber ohne weitere zeitliche Zwangsbedingungen aufgerufen werden können (vgl. Abbildung 28 b). Für andere Skills, insbesondere aus dem Bereich „Verbinden“, existieren oftmals andersartige Zusammenhänge. So müssen für den Skill „Schweißen mit elektrischem Strom“ mehrere Parameter wie Drahtgeschwindigkeit oder Stromstärke abhängig vom zeitlichen und örtlichen Verlauf der Bahnbewegung aktiviert werden.

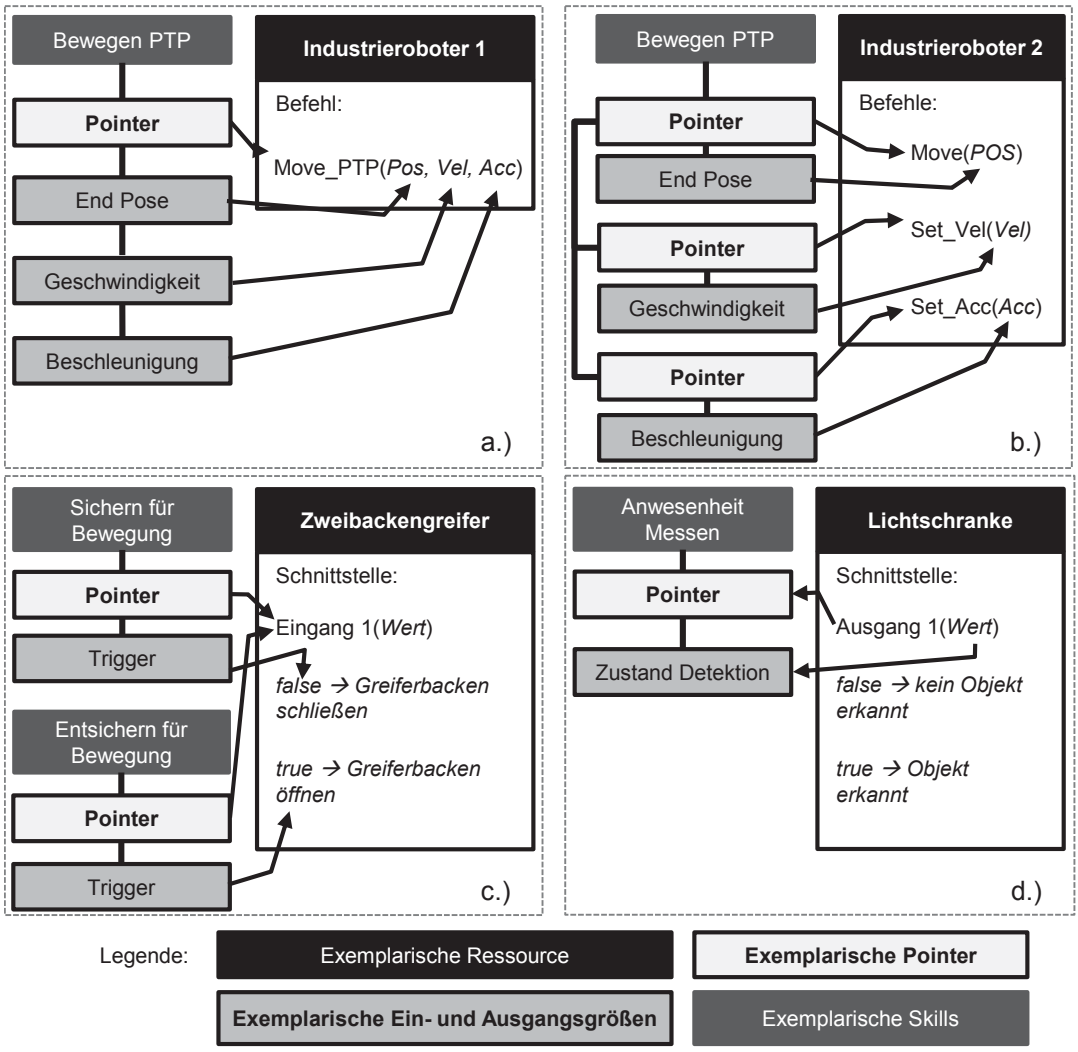


Abbildung 28: Veranschaulichung des Konzepts der Pointer für das Mapping zwischen Skills sowie Schnittstellen, Funktionen und Befehlen nach BACKHAUS & REINHART (2015)

Ein Pointer setzt sich somit aus den folgenden Bestandteilen zusammen:

- Zuordnung der Ein- und Ausgangsgrößen zu den Elementen der Befehle bzw. Funktionen inklusive notwendiger Werte für die Zuordnung und Umrechnungsfaktoren
- Notwendige Reihenfolge für den Aufruf der Befehle, Funktionen oder Schnittstellen

Die durch Eingangsgrößen beeinflussten und damit veränderbaren Eigenschaften, wie z. B. die Kinematik eines Roboters, müssen außerdem alle mit einem Startwert referenziert sein. Dieser referenzierte Zustand bildet den Ausgang für den Planungsprozess.

Typen der direkten Skills im Planungsprozess

Aus der Betrachtung der gesammelten Ressourcen und der dabei vorkommenden Skills wurden drei Typen von direkten Skills identifiziert:

- Einfache direkte Skills: Direkte Zuordnung von Befehl oder Schnittstelle zu einem Skills ohne zeitliche Abhängigkeiten.
- Komplexe direkte Skills: Zuordnung zwischen Befehl oder Schnittstelle zu einem Skill mit zeitlichen Abhängigkeiten der Ein- oder Ausgangsgrößen. Diese lassen sich wieder in einzelne Bestandteile mit jeweiligem Pointer zerlegen, haben aber eine direkte Abhängigkeit zu allen anderen Bestandteilen, welche dem lösbaren komplexen Skill zugeordnet werden können. Sie können jedoch nicht als eigene Skills angesehen werden, da sie alleinstehend keiner der fünf übergeordneten Klassen an Skills zugeordnet werden können. Ein Beispiel ist das oben genannte „Schweißen mit elektrischem Strom“, bei dem für eine Schweißbahn Gaszufuhr, Elektrodenspannung, und Drahtgeschwindigkeit zusammenhängend gesteuert werden müssen.
- Kombinierte direkte Skills: Kombination von zwei oder mehreren einfachen Skills, die ohne zeitliche Abhängigkeit mit einem gemeinsamen Pointer angesprochen werden. Diese lassen sich nicht wieder in einfache Skills mit jeweiligen Pointer zerlegen. Ein Beispiel wäre die Kopplung der Kinematik für die Bewegung mit der Kinematik des Greifers. Das Auslösen des Greifens ist also durch die Kinematik vorgegeben und kann nicht direkt angesteuert werden.

5.3.4.2 Indirekte Skills

Die Modellierung der indirekten Skills und deren Klassen orientiert sich im Wesentlichen an den Ergebnissen von SELIG (2011). Folgende übergeordnete Klassen von *Kommunikations-Skills* werden im Rahmen dieser Arbeit verwendet, die den Funktionsklassen von SELIG (2011, S. 100) entsprechen und dort beschrieben sind:

- Buskonfiguration
- Kommunikationskonfiguration
- Verbindungskonfiguration
- Busdiagnose
- Synchronisation
- Überwachung

- Redundanz
- Hotplug
- Echtzeitbits
- Übertragungsmechanismen
- Nichtechtzeitkommunikation
- Bedarfsdaten

Da jedoch in dieser Arbeit von einem konfigurierten System ausgegangen wird, werden die Kommunikations-Skills an dieser Stelle nicht weiter detailliert.

Für die *Verwaltungs-Skills* wurden folgende Klassen definiert, welche auf den übergreifenden Funktionsgruppen auf Applikationsebene nach SELIG (2011, S. 102) basieren:

- Identifikation: Darin sind alle Skills, die zur Erkennung des Gerätes dienen, zusammengefasst.
- Diagnose: Die Skills dieser Klasse ermöglichen es, die Zustände einer Ressource, wie Fehler, Warnungen und Informationen, zu identifizieren.
- Administration: Diese Klasse umfasst alle Skills, die administrativen Zwecken dienen, wie z. B. die Sprachauswahl.
- Archivierung: Die Archivierung umfasst alle Skills zur Speicherung und zum Abruf von Konfigurationen und Parametersätzen.
- Zustandsmaschine: Neben den Skills zum Zugriff und zur Anzeige der Zustandsmaschine wird im Rahmen dieser Arbeit im Gegensatz zu SELIG (2011) auch die Zustandsmaschine hinterlegt.

Im Vergleich zu direkten Skills unterscheidet sich die Struktur von indirekten Skills. Da indirekte Skills nicht direkt im Produktionsprozess eingesetzt werden können, ist keine Überprüfung notwendig und deren Eigenschaften müssen außer für die Klasse „Zustandsmaschine“ nicht beschrieben werden. Indirekte Skills werden z. B. zur Koordination von Abläufen durch das Programmiersystem eingefügt. Sie setzen sich daher nur aus einem *Namen*, einem *allgemeinen Teil*, welcher ID und Beschreibung enthält, *Ein-* und *Ausgangsgrößen* sowie einer *Zustandsmaschine* zusammen. Instanziierte Skills einer konkreten Ressource besitzen jeweils auch ein *Mapping auf die Befehle bzw. Schnittstellen* der Ressource durch Pointer.

Eine Besonderheit bildet die Klasse der Zustandsmaschine. Sind Skills dieser Klasse vorhanden, wird eine Abbildung der verwendeten Zustandsmaschine als Eigenschaft für alle Skills dieser Klasse vorausgesetzt. Die Skills, welche z. B. das Schalten zwischen den Zuständen der Zustandsmaschine ermöglichen, wer-

den direkt mit dieser Zustandsmaschine verknüpft. Die Notwendigkeit dieser Abbildung wird im Folgenden Abschnitt beschrieben.

5.3.4.3 Abhängigkeiten zwischen indirekten- und direkten Skills

Wie auch bei SELIG (2011) beschrieben, sind Verwaltungs-Skills oftmals Applikations-Skills zugehörig. An dem Beispiel eines Servoantriebs soll diese Problemstellung verdeutlicht werden. Abbildung 29 zeigt die vereinfachte Zustandsmaschine eines Servoantriebs nach PLCopen (PARKER 2013) ohne die zugehörigen Befehle für die Zustandsübergänge.

Grundsätzlich erlaubt die zur dargestellten Zustandsmaschine gehörende Ressource die zwei Bewegungsarten diskret sowie kontinuierlich und besitzt somit auch die zugehörigen Skills. Um aber eine der Bewegungsarten ausführen zu können, muss die Zustandsmaschine vom Zustand „Ausgeschaltet“ in den Zustand „Stillstand“ gebracht werden. Dazu ist ein entsprechender Verwaltungs-Skill notwendig. Im Applikations-Skill muss also auch der zugehörige Zustand in der Zustandsmaschine referenziert werden, der die Ausführung des Applikations-Skills ermöglicht. Damit kann das Programmiersystem schlussfolgern, welcher Zustand zuvor erreicht werden muss. Die Modellierung erfolgt über die Angabe des notwendigen Zustands in der Zustandsmaschine als Zwangsbedingung für den Zustandsübergang des Applikations-Skills (vgl. Zustandsmodell in Tabelle 2). Somit kann automatisch abgeleitet werden, welcher Zustand im Skill der Zustandsmaschine erreicht werden muss.

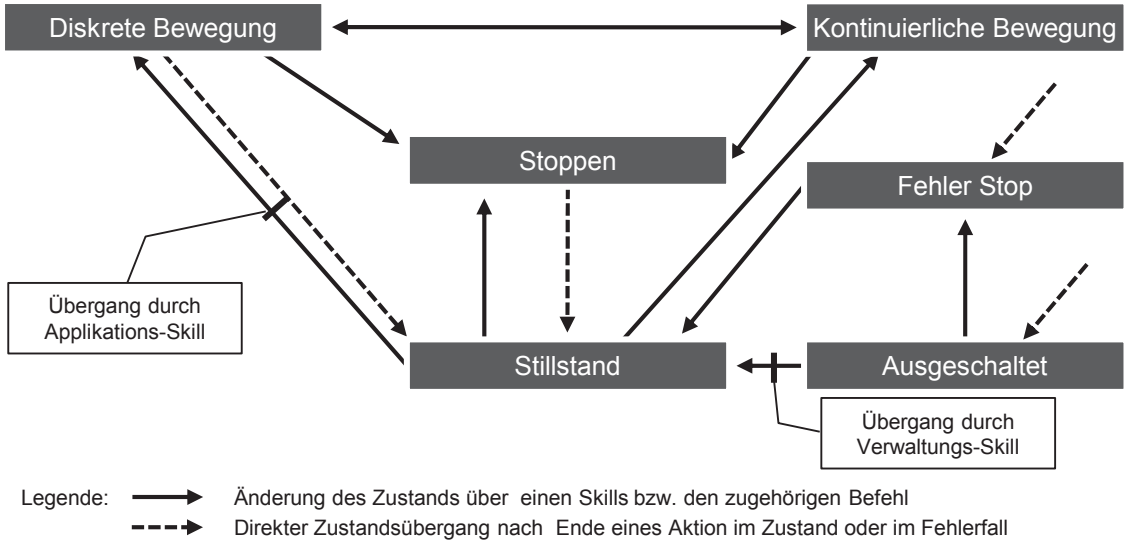


Abbildung 29: Ausschnitt der PLCopen Zustandsmaschine eines Servoantriebs nach PARKER (2013)

Vergleichbare Abhängigkeiten müssen z. B. auch in der Klasse Administration für das Wechseln der Einheiten von Ein- und Ausgangsgrößen unter Verwendung der Zustandsmaschine modelliert werden. Dadurch, dass sich alle möglichen Skills auf eine Zustandsmaschine beziehen, können keine konkurrierenden Skills unbeabsichtigt gleichzeitig ausgeführt werden, wie im oberen Fall eine kontinuierliche und diskrete Bewegung.

5.3.4.4 Verwendung der Skills im Planungsprozess

Die Modellierung der Skills inklusive deren Instanziierung für eine Ressource kann durch den Hersteller der Ressource oder mittels interner Bibliotheken des Anwenders erfolgen. Die Einordnung der Skills in eine Klassenhierarchie und die damit verbundene Vererbung von Ein- und Ausgangsgrößen sowie von Eigenschaften ermöglicht eine flexible Verwendung der Skills im Planungsprozess. Statt jede Ressource mit der spezifischen Modellierung eines Herstellers einbinden zu müssen, bilden die Skills eine gemeinsame Basis, um Ressourcen bzgl. der Verwendung im Montageprozess überprüfen zu können. Mit dem Mapping der Ein- und Ausgangsgrößen auf die Schnittstellen, Funktionen und Befehle der Ressource, bilden die Ein- und Ausgangsgrößen die Referenz, um die zugehörigen Größen zu erzeugen bzw. zu verwenden. Das Programmiersystem arbeitet mit einem definierten Stand an Skills, die in einer Hierarchie beschrieben sind. Durch die Vererbung können auch Ressourcen in die Hierarchie eingebunden werden, welche Spezialisierungen einer höheren Klasse darstellen. Dabei kann es jedoch vorkommen, dass nicht alle Eingriffsmöglichkeiten der neuen Ressource genutzt werden können. Würde z. B. ein Industrieroboter für den Skill „Geführte Lineare Bewegung“ auch die Vorgabe eines Geschwindigkeitsprofils ermöglichen, könnte dieser Freiheitsgrad nicht genutzt werden, falls das Programmiersystem einen abstrakteren Skill ohne diese Eingangsgröße verwendet. Der spezialisierte und instanziierte Skill muss in diesem Fall Default-Werte für diese Eingangsgrößen liefern.

Vor allem im Bereich der SPS-gesteuerten Montagesysteme existieren oftmals speziell kombinierte Montagekomponenten, welche aus einzelnen Prozessträgern aufgebaut sind. Ein Beispiel ist eine Vorrichtung zum Fügen von Bauteilen durch Verpressen. Diese kann in Abhängigkeit von den konkreten Anforderungen des Montageprozesses und des Bauteils sehr spezifisch aufgebaut sein, beispielweise durch eine unterschiedliche Anzahl an Antrieben. Für die spezialisierten Vorrichtungen ist es nicht zielführend, jeweils eigene Skills abzuleiten, da somit auch ein zugehöriges Programmiersystem sehr speziell wäre. Das Konzept in dieser Ar-

beit fokussiert daher flexible Montagevorrichtungen, die aus Standardkomponenten aufgebaut sind. Spezialisierte Montagekomponenten können jedoch einfach eingebunden werden, indem deren Funktionalität z. B. durch zugehörige Softwarebausteine gekapselt wird und sie somit Skills auf einer höheren Abstraktionsebene bereitstellt. Eine Entwicklung dieser Bausteine zur Abstraktion von Komponenten ist aktuell Gegenstand der Forschung und wird zum Beispiel auch von HAMMERSTINGL & REINHART (2015) vorgeschlagen.

5.3.5 Modellierung der Ressourcen

Die Beschreibung einer Ressource unterscheidet drei wesentliche Klassen für den Anwendungsfall der aufgabenorientierten Programmierung, die auf einer einheitlichen Grundstruktur basieren. Diese sind steuerbare Ressourcen (z. B. Industrieroboter oder Greifer), Steuerungen bzw. Zielsysteme (z. B. Robotersteuerung oder SPS), sowie Quellen und Senken. Für alle Elemente existiert eine übergreifende Beschreibungsstruktur, die aber für die einzelnen Typen erweitert wird. Für den Inhalt der Beschreibung wurden Teile des Konzepts von KRUG (2013) verwendet und die dortigen Inhalte für die aufgabenorientierte Programmierung erweitert. Abbildung 30 zeigt das Klassendiagramm der Klasse Ressource und deren Spezialisierungen.

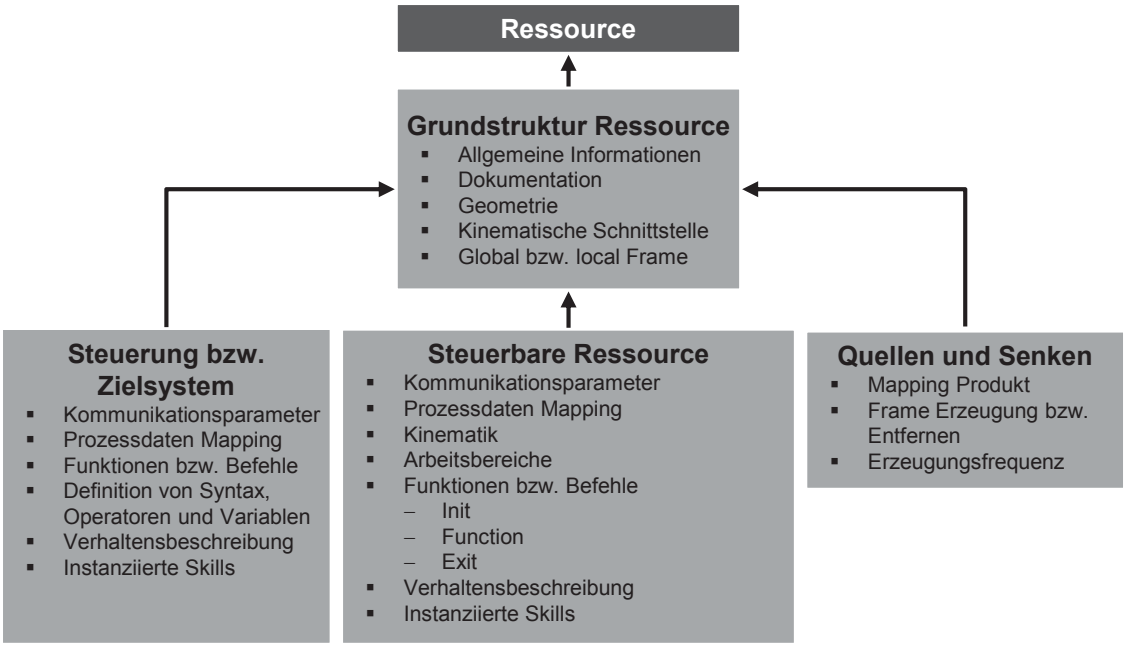


Abbildung 30: Klassendigramm der Klasse Ressource

Die *Grundstruktur* einer Ressource besteht aus deren allgemeinen Informationen, wie z. B. ID und Name, der Geometrie, der Dokumentation wie z. B. Handbü-

cher, der kinematischen Schnittstelle und deren global oder local Frame. Der global bzw. local Frame definiert die räumliche Position und Orientierung (x, y, z, z, x‘, z‘‘) in Bezug zum Weltkoordinatensystem bzw. zu einer anderen Ressource. Dessen Wert wird typischerweise erst mit der Instanziierung im Umweltmodell festgelegt. Mit der kinematischen Schnittstelle können, repräsentiert durch ein Frame und ein kinematisches Gelenk (typischerweise ein starres Gelenk), die kinematischen Verbindungen zweier Ressourcen definiert werden. Mit der Grundstruktur ist es möglich, z. B. Werkstückträger oder Schutzzäune abzubilden.

Eine *Steuerung bzw. ein Zielsystem* ist nicht direkt im Produktionsprozess beteiligt und hat die zusätzlichen Elemente Kommunikationsparameter, Prozessdaten Mapping, Definition von Befehlen, Syntax, Operatoren und Variablen, Verhaltensbeschreibung sowie instanziierte Skills. Kommunikationsparameter stellen, wie in KRUG (2013) definiert, eine Beschreibung der Kommunikation in Abhängigkeit der verwendeten Technologie (Profibus, Ethernet, RS232, etc.) in der jeweiligen Beschreibungssprache (z. B. GSD, XDD) dar. Auch das Prozessdaten Mapping, d. h. die Beschreibung der Schnittstellen und deren Zuordnung zu Schnittstellen von anderen Ressourcen im Montagesystem, wurde aus dem Konzept von KRUG (2013) übernommen. Das Verhalten der Steuerung wird durch eine zugeordnete Beschreibung oder ein Beschreibungsmodell abgebildet. Zusätzlich ist auch der Zusammenhang zwischen den Schnittstellen und Befehlen und der Verhaltensbeschreibung abzubilden. Die Modellierung von Verhalten wurde schon in einer Vielzahl von Arbeiten betrachtet (LINDWORSKY 2011; SCHAICH 2001; BERGERT ET AL. 2010). Bestehende Konzepte können daher für diese Arbeit übernommen werden. Die instanziierten Skills beinhalten die Zuordnung der übergreifend gültigen Skills zu den Befehlen und Deklarationen des Zielsystems sowie dessen Verhaltensmodell. Die Allgemeinen Informationen des Zielsystems umfassen Elemente wie Versionsnummer, CPU, Taktfrequenz und Speicher.

Eine s*teuerbare Ressource* wird über Kommunikationsschnittstellen angesteuert oder über ein Zielsystem programmiert. Sie wird durch die zusätzlichen Elemente Kommunikationsparameter, Prozessdaten Mapping, Arbeitsbereiche, Verhaltens-beschreibung, Funktionen bzw. Befehle und instanziierte Skills beschrieben. Mit der Kinematik werden die statischen und dynamischen Eigenschaften von Ressourcen abgebildet. Sie beschreibt die Geometrie in Zusammenhang mit den zeitabhängigen Aspekten der Bewegung (RIESELER 1992) mit Größen wie Position, Geschwindigkeit, Beschleunigung und Zeit (HAUN 2007). Das Informations-

modell muss daher alle dafür notwendigen Parameter der jeweiligen Ressource enthalten. Die Modellierung der Kinematik kann darüber hinaus Hilfskoordinatensysteme enthalten, wie sie z. B. im Bereich der Robotik für den Tool-Center-Point (TCP) eingesetzt werden (KRUG 2013). Die Kommunikationsparameter und das Prozessdaten Mapping werden wie in der Ressourcenklasse Zielsystem beschrieben. Arbeitsbereiche definieren die Geometrie des Arbeitsraums oder Arbeitspunktes einer Ressource bezogen auf deren Geometrie. Je nach gewünschtem Detaillierungsgrad kann ein Bounding Box Ansatz oder eine Annäherung über geometrische Körper wie Kugeln oder Halbkugeln erfolgen. Wie bei einer Virtuellen Inbetriebnahme, muss für eine Ressource auch das zugehörige Verhalten der Ressource beschrieben werden. Nur damit ist z. B. die Überprüfung möglich, in welche Bewegungsrichtungen sich ein Förderband bewegen kann. Die Verhaltensbeschreibung muss zusätzlich auf die Geometrie- und Kinematikbeschreibung der Ressource abgebildet werden (BERGERT ET AL. 2010). Andernfalls wäre nicht explizit modelliert, welche Elemente eines Förderbandes sich wann in welche Richtung bewegen. Werden von einer steuerbaren Ressource Befehle, Funktionen oder Funktionsbausteine für die Verarbeitung in einem Zielsystem zur Verfügung gestellt, so müssen diese beschrieben und mit den instanziierten Skills verknüpft sein. Diese werden in den drei Bereichen Init, Function und Exit hinterlegt. Im Bereich Init werden z. B. Funktionsdefinitionen oder notwendige Initialisierungen abgelegt. Wären diese nicht getrennt abgelegt, so könnte der Postprozessor nicht schlussfolgern, welche Initialisierungen und Definitionen notwendig sind. Der Bereich Function stellt die eigentlichen Funktionalitäten der Ressource dar, welche mit den Skills verknüpft werden. Der Bereich Exit umfasst im Gegensatz zur Initialisierung alle Befehle, Funktionen bzw. Routinen, die am Ende einer Programmausführung notwendig sind, um evtl. laufende Funktionen zu beenden. Für programmierbare Ressourcen, wie z. B. Industrieroboter, ist das zugehörige Zielsystem zu referenzieren und die direkt zur Ressource gehörenden Befehle abzulegen sowie mit den Skills zu verknüpfen. Diese Funktionen, Schnittstellen oder Befehle müssen zudem auf die Verhaltensbeschreibung der Ressource abgebildet werden. Die Skills werden für die jeweilige Ressource mit deren Eigenschaften instanziiert und mit den zugehörigen Befehlen, Funktionen bzw. Schnittstellen aus dem Bereich Function sowie der Verhaltensbeschreibung verknüpft.

Quellen und Senken stellen sogenannte Entwurfsmetaphern bzw. Hilfsmittel für die Simulation dar und ermöglichen die Abbildung von Schnittstellen, an denen Produkte in das System gebracht oder aus diesem geführt werden (LACOUR

2012). Sie sind daher mit Produkten verknüpft, die in vorgegebener zeitliche Frequenz und vorgegebenem Frame erzeugt werden.

5.3.6 Aufgabenmodell

Abhängig vom Inhalt des Aufgabenmodells, d. h. welche Informationen zur Beschreibung der Aufgabe vorliegen, ergeben sich die notwendigen Funktionalitäten des Programmiersystems, da dieses die Lücke zwischen dem Aufgabenmodell und dem Steuerungsprogramm des Montagesystems füllen muss. Im Rahmen dieser Arbeit wurde ein hierarchischer Ansatz gewählt, der eine Anpassung des Aufgabenmodells an die Randbedingungen im Unternehmen ermöglicht. Die Ebenen des Aufgabenmodells geben somit auch die Ebenen im Rahmen des Planungsprozesses des Programmiersystems vor. Für die Entwicklung der Ebenen im Aufgabenmodell wurden bestehende Ansätze angepasst (DILLMANN & HUCK 1991; MOSEMANN 2000; THOMAS ET AL. 2013; SCHOU ET AL. 2013). Es wurden die nachfolgend beschriebenen Ebenen abgeleitet. Da nur in den Ebenen 1 bis 4 eine Abstraktion der Aufgabe stattfindet, stehen nur diese als Möglichkeit zur Beschreibung durch den Endanwender bereit. Abbildung 31 zeigt ein Beispiel für die Ebenen des Aufgabenmodells mit den notwendigen Funktionalitäten auf Seiten des Programmiersystems.

Primärprozessebene (1): Nur die Füge (Primär-) Prozesse sowie der Anfangs- und Endzustand der zugehörigen Produkte werden beschrieben, nicht deren Reihenfolge. Eine Ausnahme bilden Sekundärprozesse, die der Qualitätssicherung dienen und daher auch auf dieser Ebene beschrieben werden. Diese sind noch lösungsneutral bzgl. des Montagesystems.

Montagesequenzebene (2): Neben den Primärprozessen wird in dieser Ebene auch die Reihenfolge der Primärprozesse definiert. Die Sequenzbeschreibung kann mögliche parallele Prozessschritte beinhalten und ist daher mit einem Montagevorranggraphen vergleichbar.

Sekundärprozessebene (3): In dieser Ebene sind neben den Primärprozessen die sich aus den Randbedingungen der Anlage ergebenden Sekundärprozesse in die Montagesequenz eingegliedert. Ab dieser Ebene ist die Aufgabenbeschreibung somit nicht mehr komplett unabhängig, sondern in ihrer Sequenz für das jeweilige Montagesystem angepasst.

Skillebene (4): Die Skillebene beschreibt die Aufgabe mit einer Sequenz von Skills, ohne deren Eingangsgrößen bzw. Parameter vorzugeben. Die Sequenz

entspricht nicht zwangsläufig der endgültigen Sequenz, da z. B. die Anzahl der Wiederholungen noch nicht spezifiziert ist. Ein Beispiel ist der Vorgang eines Handhabungsprozesses, der sich aus einem Ablauf typischer Funktionen zusammensetzt, wobei die Anzahl der Bewegungen von der Umgebung abhängt.

Parameterebene (5): Die Parameterebene beschreibt die Montageoperation durch Skills inklusive der Vorgabe aller notwendigen Eingangsgrößen. Diese Ebene entspricht somit der Programmierung mit einer abstrakten Programmiersprache.

Codeebene (6): Die Codeebene stellt den herstellerspezifischen Code für die Zielsysteme im Montagesystem zur Ansteuerung der Ressourcen dar und repräsentiert somit auch das Ziel der Planung im Programmiersystem. Diese Ebene ist somit nicht direkt Bestandteil einer Aufgabenbeschreibung.

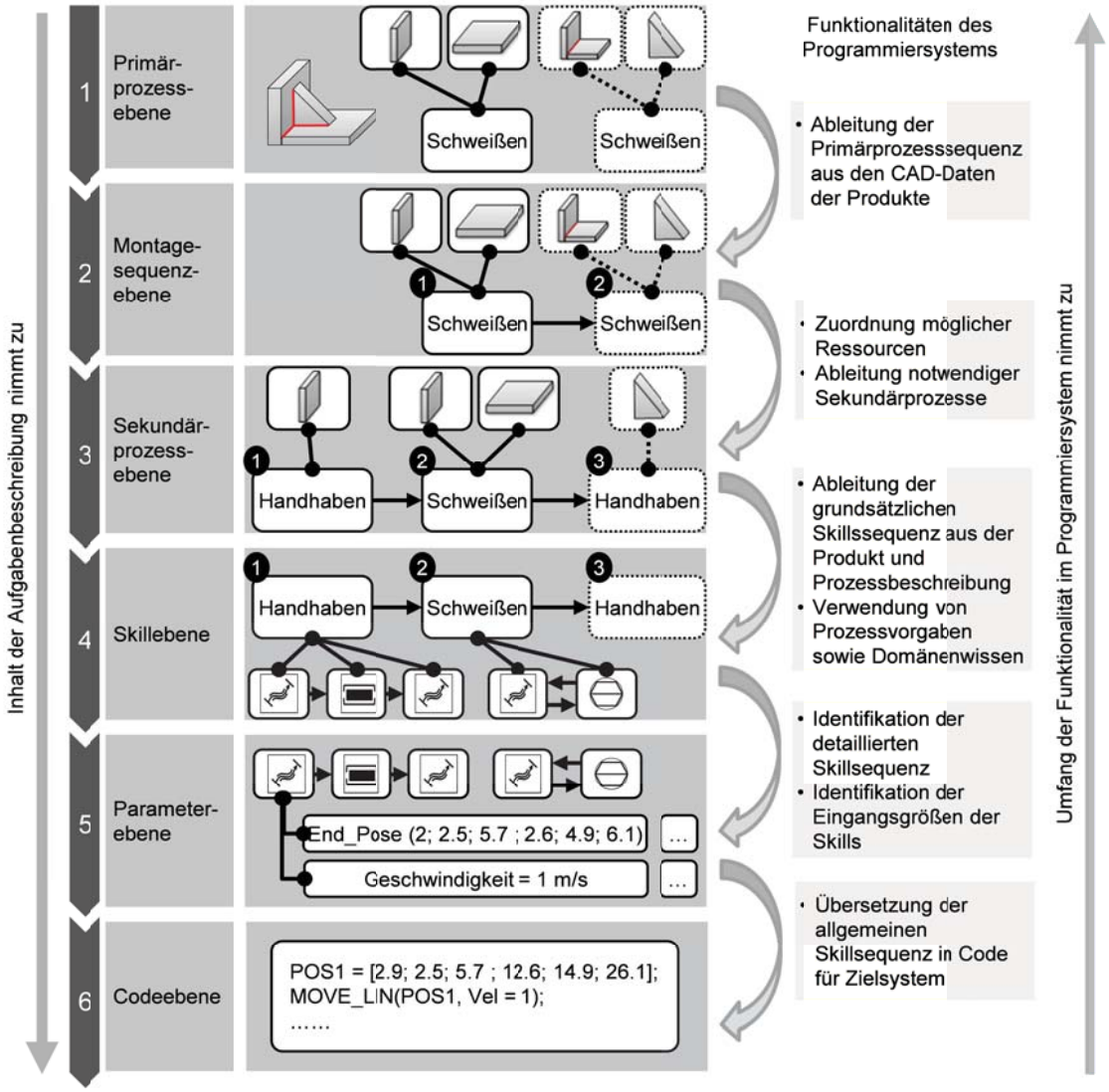


Abbildung 31: Ebenen des Aufgabenmodells und notwendige Funktionalitäten im Programmiersystem

5.3.7 Umweltmodell

Das Umweltmodell beschreibt den aktuellen Zustand des Montagesystems und beinhaltet somit die Beschreibung aller Ressourcen, deren Skills sowie deren Beziehungen zueinander. Im Folgenden werden zunächst die Inhalte des Umweltmodells beschrieben, bevor in einem anschließenden Abschnitt eine Methode für dessen teilautomatische Generierung vorgestellt wird.

Ressourcen: Das Umweltmodell beinhaltet alle Ressourcen inklusive der kinematischen und kommunikationstechnischen Beziehungen über die jeweiligen Schnittstellen sowie deren Frame, bezogen auf eine andere Ressource oder ein Weltkoordinatensystem.

Umweltbedingungen: In diesem Bereich werden alle Bedingungen zusammengefasst, welche die Umwelt des Montagesystems beschreiben. Beispiele sind Lichtquellen (z. B. Umgebungslicht, Punktlicht oder Flächenlicht), Raumtemperatur und Luftfeuchtigkeit. Diese Größen spielen für den Einsatz einzelner Ressourcen, wie z. B. einem Kamerasystem, eine besondere Rolle.

Simulationsspezifische Inhalte: Um im Rahmen des Programmiersystem ein Simulationsmodell generieren zu können, sind je nach Simulationssystem teilweise weitere Größen notwendig. Diese umfassen z. B. Templates für das Zellenmodell.

5.3.8 Integration des Informationsmodells in ein Beschreibungsformat

Die in den vorangegangenen Abschnitten beschriebenen Bestandteile des Informationsmodells sind prinzipiell unabhängig von der konkreten Umsetzung in einer Modellierungssprache bzw. einem Beschreibungsformat. Die Verwendung von standardisierten Beschreibungsformaten bietet jedoch Vorteile, da die formale Korrektheit und Vollständigkeit sichergestellt werden (KRUG 2013). Das Informationsmodell hat in seinen Modellbestandteilen, wie Ressourcen und Prozessen, überschneidende Informationsbereiche. So werden z. B. in beiden Bereichen geometrische Informationen beschrieben. Bei der Integration in ein Beschreibungsformat sollten jedoch für gleiche Informationsbereiche auch die gleichen Beschreibungsmittel verwendet werden. Daher wurden alle Elemente des Informationsmodells zunächst analysiert und die vorhandenen Informationsbereiche identifiziert. KRUG (2013) verwendet in seiner Arbeit sechs Informationsbereiche, welche für diese Arbeit erweitert werden müssen. Folgende Informationsbe-

reiche wurden für den Anwendungsfall der aufgabenorientierten Programmierung abgegrenzt:

1. *Hilfe und Dokumentation*: Dieser Teilbereich beinhaltet wie bei KRUG (2013) weiterführende Informationen der Ressourcen wie Handbücher, die bei manuellen Anpassungen der erzeugten Programme relevant sein können.
2. *Allgemeine Informationen*: Diese umfassen Informationen wie Hersteller, Namen von Ressourcen (KRUG 2013) und weitere Beschreibungselemente wie die Typbezeichnung. Neben Einzelelementen sind auch Struktur und Hierarchie abzubilden.
3. *Geometrie- und Kinematikbeschreibung*: In diesem Informationsbereich werden geometrische Größen und kinematische Größen wie Abmessungen von Produkten, Gelenke, Verbindungsglieder, Achsen und Koordinatensysteme zusammengefasst (KRUG 2013).
4. *Funktionen, Befehle und Syntax*: Dieser Bereich bildet die Befehle, Funktionen und den Syntax der Ressourcen ab.
5. *Mapping*: Beschreibt neben der Verknüpfung der Prozessdaten von verschiedenen Ressourcen über die Kommunikationssysteme (KRUG 2013) auch die Verknüpfung zwischen den Ein- und Ausgangsgrößen von Skills und den Befehlen einer spezifischen Ressource (Pointer) sowie die Abhängigkeiten der Verhaltensmodelle zur Geometrie und Kinematikbeschreibung einer Ressource.
6. *Echtzeitkommunikation*: In diesem Teilbereich wird die Echtzeitkommunikation festgelegt (KRUG 2013).
7. *Abläufe und Sequenzen*: Diese spezifizieren die Abläufe bzw. Sequenzen von Prozessen bzw. Skills.
8. *Zustandsmaschinen*: Zur Beschreibung der Skills mit deren zugehörigen Zustandsmaschine.
9. *Steuerungs- und Maschinenverhalten*: Bilden das Verhalten von Ressourcen ab.

Für den Anwendungsfall der aufgabenorientierten Programmierung können aus den beschriebenen Modellinhalten folgende Anforderungen an das Beschreibungsformat abgeleitet werden. Zunächst muss das Format maschinenlesbar sein, um eine Verarbeitung im Programmiersystem zu ermöglichen. Zur Abbildung der notwendigen geometrischen und kinematischen Bestandteile, aber auch der Verhaltensmodelle, müssen entsprechende Beschreibungsmöglichkeiten vorliegen. Die Beschreibung der Ressourcen erfordert des Weiteren die Modellierung

technischer Details, wie z. B. des globalen Frames. Zur Abbildung der Skills und deren Hierarchie, müssen darüber hinaus auch semantische Informationen und Hierarchien abgebildet werden können.

Aus den gestellten Anforderungen kann gefolgert werden, dass graphisch orientierte Modellierungssprachen wie UML oder SysML nicht geeignet sind. Austauschformate, wie sie auch heute für den Austausch von Daten zwischen Entwicklungswerkzeugen zum Einsatz kommen, sind hingegen geeignet. Da in dieser Arbeit ein breites Feld an Modellinhalten notwendig ist, können nicht einzelne Standards wie PLCopen, CAEX oder STEP verwendet werden. Wie bei KRUG (2013) erfolgt daher auch in dieser Arbeit die Integration in den Engineeringstandard AutomationML. Dieses XML basiertes Format bietet neben der Maschinenlesbarkeit auch den Vorteil der Referenzierung weiterer Standards und ist damit erweiterbar. Die Informationsbereiche des Informationsmodells mit den zugehörigen Beschreibungsstandards sind in Abbildung 32 dargestellt.

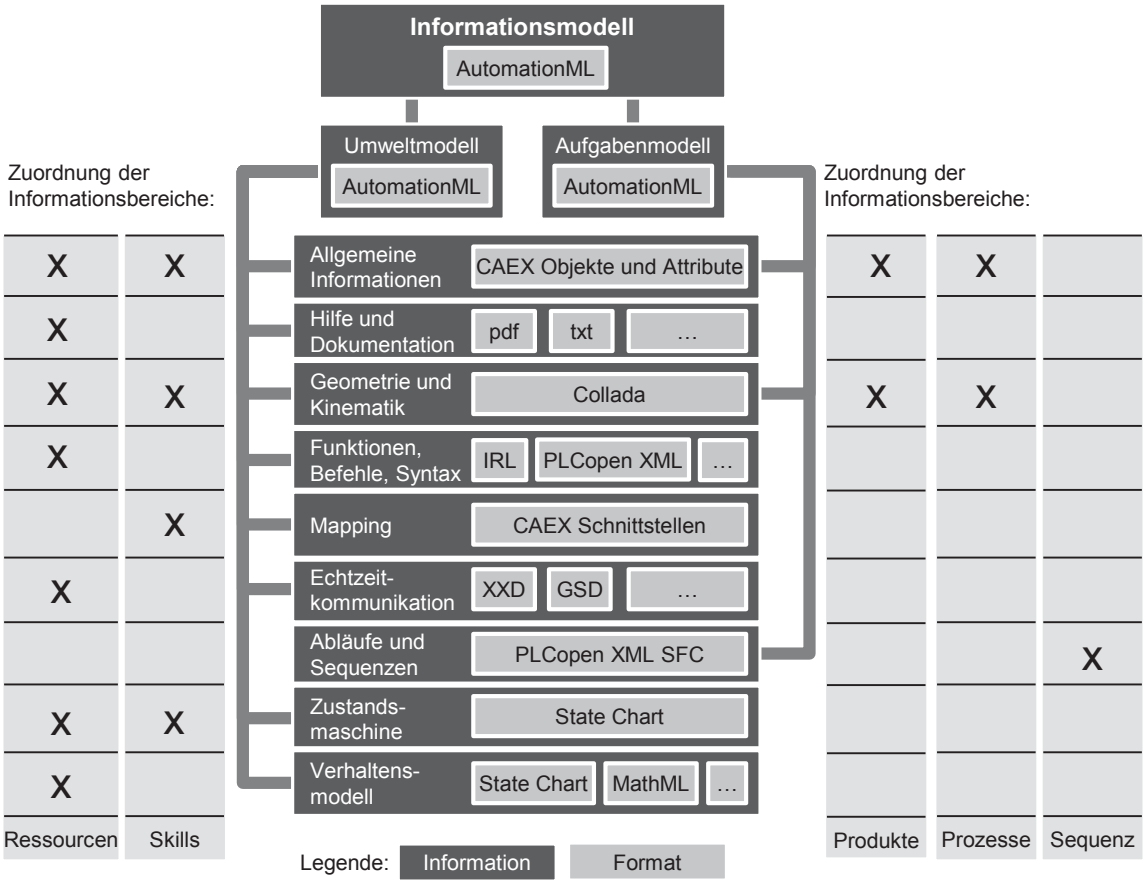


Abbildung 32: Informationsarten und Beschreibungsformate des Informationsmodells

Da die Informationsbereiche *allgemeine Informationen, Hilfe und Dokumentation* sowie *Geometrie und Kinematik* weitestgehend von KRUG (2013) übernommen wurden, werden deren vorgeschlagene Beschreibungsformate an dieser Stelle nur bezüglich der Erweiterungen für Sills erläutert.

Allgemeine Funktionen: Diese Bereich umfasst auch die Abbildung der Skills als Objekte, deren Modellierung kurz betrachtet wird. Skills werden in der Instanz eines Montagesystems als Objekte einer Ressource mit entsprechenden Attributen und CAEX-Interfaces modelliert. Die übergeordnete Skill-Hierarchie wird dagegen als Rollenklassenbibliothek in AutomationML modelliert. Die Rolle der zugehörigen Skill-Klasse wird dem instanziierten Skill zugewiesen. Wenn das Programmiersystem auf einen spezifischen Stand der Skill-Hierarchie ausgelegt ist, kann automatisch bzgl. der Verwendbarkeit des Skills bei der Programmierung geschlossen werden.

Funktionen, Befehle und Syntax: Je nach Zielsystem wird hier IRL für Robotersteuerungen oder PLCopen XML Funktionsbausteine für Speicherprogrammierbare Steuerungen oder auch der steuerungsspezifischer Code für steuerungsintegrierte Systeme, wie z. B. Industrieroboter hinterlegt. Da die Steuerung eines Roboters durch die Robotersteuerung eines anderen Herstellers die Ausnahme darstellt, kann damit Aufwand in der Übersetzung reduziert werden. Entsprechend werden also z. B. die Funktionen bzw. Befehle eines KUKA Roboters in der KUKA Robot Language (KRL) oder die eines ABB Roboters in Rapid hinterlegt. Da Komponenten teilweise sowohl in Robotersteuerungen als auch SPS integriert werden können, ist es teilweise notwendig, sowohl eine Beschreibung in IRL, als auch in PLCopen zu hinterlegen. Für Syntaxbeschreibungen von Steuerungen werden XML-Dokumente verwendet.

Mapping: Neben dem bei KRUG (2013) definierten Mapping der Prozessdaten können sowohl die Pointer der Skills zum Verknüpfen der Ein- und Ausgangsgrößen von Skills mit den Befehlen und Funktionsbausteinen einer Ressource, als auch die Verbindung der Verhaltensmodelle mit der Geometrie und Kinematik modelliert werden. Insbesondere bei der Abbildung der Pointer von Skills müssen umfangreiche Informationen hinterlegt werden. Für das Mapping wurde eine neue Interface Klasse (SkillMapping Interface) definiert, mit der die notwendigen internen Verbindungen im AutomationML Format erzeugt werden können. Durch Attribute der Interface Klasse erfolgt das detaillierte Mapping in die Bestandteile der Funktionsbeschreibung, d. h. damit wird definiert, wie die Eingangsgrößen des Skills in der Funktionsbeschreibung benannt sind.

Abläufe und Sequenzen: Um Prozessabläufe und Sequenzen zu beschreiben, wird die Verwendung von PLCopen XML SFC (Sequential Function Charts) vorgeschlagen. Mit diesem können sowohl vom Nutzer definierte Abläufe, als auch aus CAD-Daten abgeleitete Prozessreihenfolgen beschrieben werden. Prozesse und Skills werden grundsätzlich als Steps innerhalb des SFC modelliert. Skills können jedoch auch als Übergangsbedingung an Transitionen referenziert werden. Typischerweise besteht die Verwendung eine Skills nur aus einem Step im SFC. Für die Modellierung von Skills deren Zustandsmaschine einen zweiten Trigger für die Rückkehr in den Ausgangszustand vorsieht, werden zwei Steps je Skill verwendet. Beispielsweise wäre ein Step für das Sichern eines Bauteils zuständig, der zweite Skill-Step würde das Sichern wieder beenden. Die Abbildung des Skills im SFC ändert sich somit in Abhängigkeit von dessen Zustandsmaschine.

Zustandsmodelle: Um Zustandsmaschinen abzubilden, eignen sich die im AutomationML integrierten State Charts.

Verhaltensmodelle: Da bei der Modellierung des Verhaltens neben diskreten Zuständen auch kontinuierliche Eigenschaften von Ressourcen modelliert werden müssen, werden neben PLCopen XML State Charts auch MathML Beschreibungen verwendet. Ein Beispiel zur Verhaltensmodellierung mit MathML wird z. B. in BERGERT ET AL. (2010) beschrieben.

5.3.9 Informationsverarbeitung bei der Generierung von Umwelt- und Aufgabenmodell

Generierung des Umweltmodells

Im Rahmen dieser Arbeit wird davon ausgegangen, dass die kommunikationstechnische Konfiguration des Montagesystems bekannt ist. Da jedoch einige Informationen der Konfiguration, wie z. B. das Mapping der Prozessdaten auch direkt für das Umweltmodell der Programmierung relevant sind, wurde auf das Konzept von KRUG (2013) aufgesetzt. Dessen Methode für die Informationsverarbeitung betrachtet als Zielsystem Offline-Programmiersysteme, musste für den Anwendungsfall in dieser Arbeit jedoch angepasst werden. Insbesondere wurde das Layout nicht beachtet. Abbildung 33 zeigt den Überblick des angepassten Vorgehens.

Die Ressourcenbeschreibungen werden durch einen um die Skills erweiterten Konfigurationsmanager mit den zugehörigen Transitionstreibern zu einem er-

weiterten Zustandsmodell verarbeitet. Die Positionen und Orientierungen der Ressourcen sowie Umweltbeschreibung sind zusätzliche Eingangsgrößen in das aufgabenorientierte Programmiersystem und werden mit dem erweiterten Zustandsmodell über das Datenmanagement zum Umweltmodell zusammengefasst.

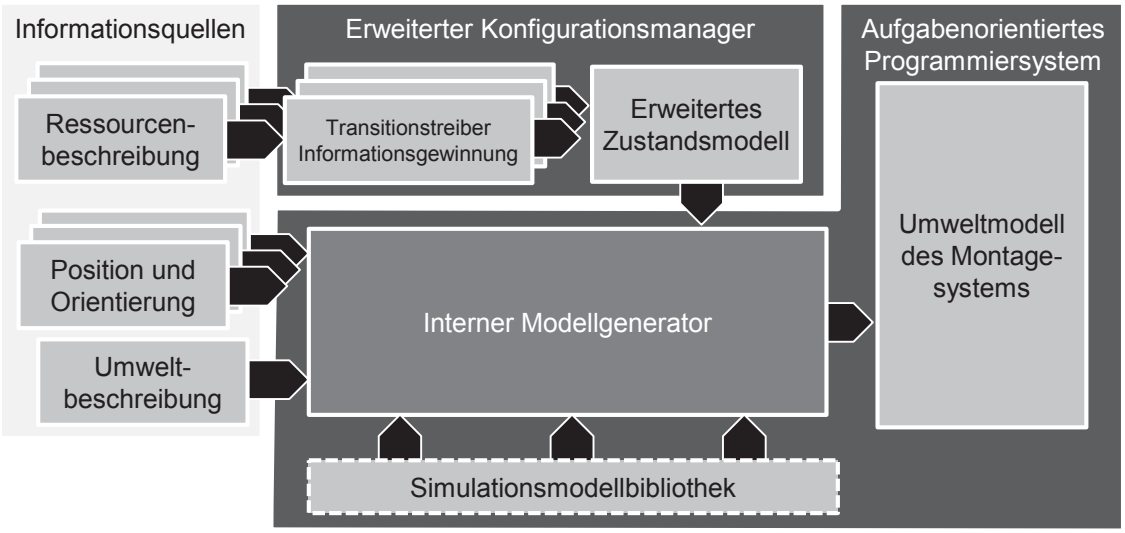


Abbildung 33: Übertragungswege und Informationsverarbeitung bei der Generierung des Umweltmodells

Somit wird der in der Übersicht des aufgabenorientierten Programmiersystems beschriebene Modellgenerator (vgl. Abbildung 17) durch zwei Bestandteile instanziiert. Insgesamt gibt es drei wesentliche Informationsquellen. Dies sind Ressourcenbeschreibungen, Positionen und Orientierungen sowie Umweltbeschreibungen. Ressourcenbeschreibungen können vom Hersteller der Ressourcen zur Verfügung gestellt bzw. automatisch aus bestehenden Engineeringdaten generiert werden. Alternativ legt sich der Anwender des Programmiersystems eine Bibliothek der von ihm verwendeten Elemente an. Das Bereitstellen von Modellen durch die Hersteller wird z. B. auch für die Virtuelle Inbetriebnahme vorgeschlagen (KIEFER ET AL. 2009). Eine Methode zur automatischen Generierung aus Engineeringdaten wird z. B. von KUFNER (2012) vorgestellt. Die Ressourcenbeschreibungen müssen die in Abschnitt 5.2 definierten Inhalte umfassen, die verwendeten Standards können auf Grund der Transitionstreiber beliebig gewählt werden. Neben den Ressourcenbeschreibungen sind die einzelnen Positionen und Orientierungen bzgl. einem Weltkoordinatensystem oder einer Ressource eine weitere Informationsquelle. Die Positionen und Orientierungen der Ressourcen im Layout müssen Referenzen zu den jeweiligen Ressourcen, z. B. über deren IDs, aufweisen. Um die Positionen und Orientierungen aller Ressourcen zu erhalten, sind mehrere Wege möglich, die nachfolgend kurz skizziert

werden sollen. Existieren 3D-Layoutpläne, so können diese verwendet werden, um Positionen und Orientierungen direkt abzuleiten. VDA (2010) gibt dazu eine Übersicht über typische Datenformate im Bereich der Layoutbeschreibungen. Eine Referenzierung der zugehörigen Ressourcenbeschreibung für jedes Objekt im Layout kann jedoch manuell notwendig sein. Liegen keine Eingangsdaten vor, so können, falls die reale Anlage vorhanden ist, die zugehörigen Positionen und Orientierungen der Ressourcen z. B. mittels 3D-Laserscanner ermittelt werden. Beispiele für eine Anwendung werden von BRACHT ET AL. (2009, S. 249–336) oder RIBO & BRANDNER (2005) beschrieben.

Liegen weder Eingangsdaten aus der Planung noch eine reale Anlage vor, werden die Ressourcen mit Dummy-Werten initialisiert und der Nutzer muss diese erst im Programmiersystem positionieren und orientieren, bevor er mit diesen arbeitet. Er legt somit auch das Layout der Anlage fest. Bei übereinstimmenden Lagen von Frames der kinematischen Schnittstellen von zwei Ressourcen werden diese entsprechend dem Gelenktyp durch den Modellgenerator automatisch verbunden. Den letzten Teilbereich der Informationsquellen stellt die Umweltbeschreibung, wie Lichtquellen oder Umgebungstemperaturen, dar. Diese Informationen müssen, wenn diese nicht im Engineeringprozess dokumentiert wurden, zusätzlich beschrieben werden. Um die Integration von Ressourcenbeschreibungen zu vereinfachen und sicherzustellen, dass auch jeweils aktuelle Versionen in das Umweltmodell geladen werden, ist die Kopplung des Programmiersystems zu einem PDM/PLM-System zielführend. Über eine bidirektionale Schnittelle können die Beschreibungen einfach eingelesen werden und deren Aktualität vor der Programmierung überprüft werden.

Eine weitere mögliche interne Informationsquelle stellt eine Simulationsbibliothek für die verwendeten Simulationssysteme dar. Eine Vielzahl an Simulationssystemen verwendet nicht direkt verarbeitbare Datenformate bei der Speicherung der Modelle. Da es somit nicht bei allen Simulationssystemen möglich ist, die Modelle der Ressourcen aus der Ressourcenbeschreibung abzuleiten, wird eine Bibliothek der verwendeten Ressourcen vorgehalten. Über Name und Beschreibung können die Simulationsmodelle zugeordnet und im Umweltmodell referenziert werden.

Die Modellierung der Umwelt kann damit weitestgehend automatisiert erfolgen. Der Aufwand für den Nutzer bei der Modellierung des Montagesystems entfällt somit fast vollständig. Dies ist jedoch auch von den verfügbaren Daten aus der Planung abhängig.

Instanziierung des Aufgabenmodells

Die Instanziierung des Aufgabenmodells und die damit verbundene Informationsverarbeitung erfolgen nicht nach einem fest vorgegebenen Schema, da das Aufgabenmodell je nach gewählter Ebene andere Informationen beinhaltet. Damit kann das Aufgabenmodell an die Anforderungen der Nutzer und der Unternehmensrandbedingungen angepasst werden. Die Bestandteile des Vorgehens orientieren sich an den Ebenen des Aufgabenmodells und sind in Abbildung 34 dargestellt. Die Informationsverarbeitung wird für die einzelnen Ebenen im Folgenden beschrieben.

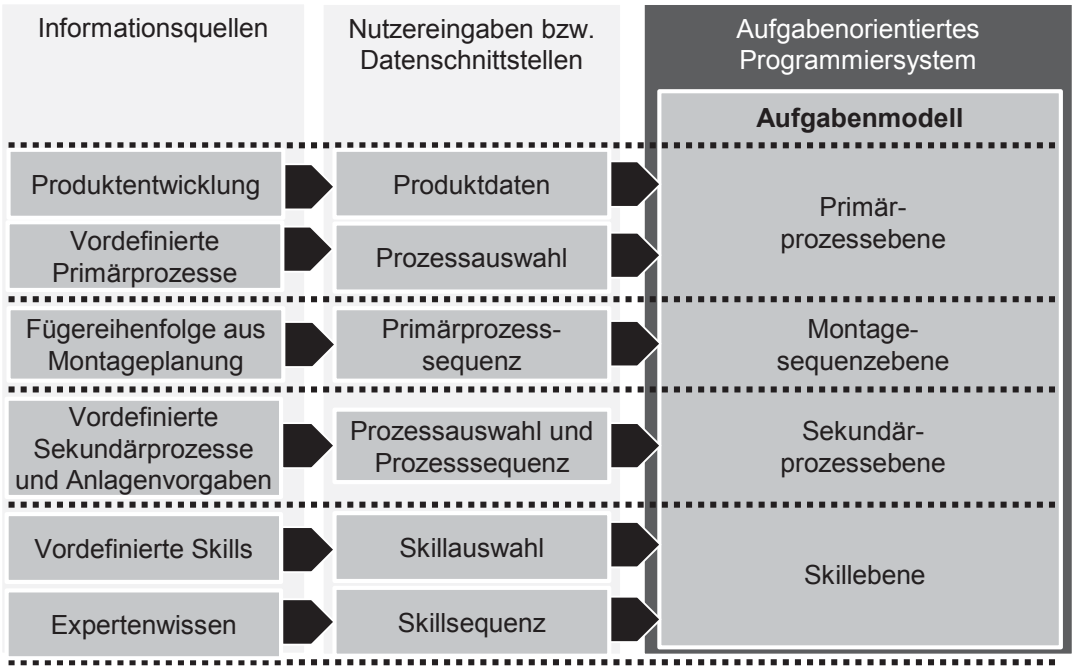


Abbildung 34: Informationsverarbeitung bei der Generierung des Aufgabenmodells

Für die Eingabe der Produktinformationen in der Primärprozessebene können die bestehenden Daten aus der Produktentwicklung verwendet werden, welche entweder durch den Nutzer vorgegeben oder direkt über Datenschnittstellen eingelesen werden. Durch die Schnittstellen können auch etwaige Konvertierungen der Daten erfolgen. Die Auswahl der notwendigen Prozesse sowie deren Verknüpfung mit den Produktdaten, können entweder durch den Nutzer erfolgen oder als zusammengehörendes Datenmodell aus dem Engineering abgeleitet werden. Auch hier ist die Verknüpfung mit PLM und PDM-Systemen zu empfehlen. In beiden Fällen sind vorab durch Experten definierte Prozesse zu verwenden. Die in der Primärprozessebene zusätzlich vorgegebene Sequenz der Primärprozesse

kann wiederum aus der Montageplanung bzw. dem Engineering verwendet oder direkt durch den Nutzer beschrieben werden. Sind zusätzlich die Sekundärprozesse und deren Sequenz definiert, erfolgt die Beschreibung der Aufgabe im Normalfall für ein spezifisches Montagesystem.

Die Prozessparameter und deren Reihenfolge wurden oftmals schon im Engineeringprozess beschrieben und können direkt verwendet werden. Liegen keine Informationen vor, ist jedoch auch eine Beschreibung durch den Nutzer möglich. Erfolgt die Aufgabenbeschreibung auf Skillebene, wird die Eingabe typischerweise durch einen Nutzer mit dessen Expertenwissen definiert. Die Skills werden dem Nutzer in der Oberfläche des Programmiersystems zur Auswahl vorgeschlagen.

Das beschriebene Vorgehen zur Informationsverarbeitung zeigt, dass durch das hierarchische Aufgabenmodell bestehende Daten aus dem Engineering- und Montageplanungsprozess flexibel als Eingangsgrößen für die Programmierung eingebunden werden können. Damit ist deren Wiederverwendung möglich und der Aufwand zum Einsatz des aufgabenorientierten Programmiersystems wird reduziert. Das zugehörige Informationsmodell beschreibt die notwendigen Bestandteile für die Durchführung einer aufgabenorientierten Programmierung.

5.4 Konzeption eines modularen Planungsmoduls

Die Architektur des Planungsmoduls wurde in ihren Grundzügen in BACKHAUS & REINHART (2014) vorgestellt. Im Folgenden werden wesentlichen Elemente und Zusammenhänge zusammengefasst. Im Rahmen dieser Arbeit wurden nur übergreifende Funktionalitäten des Planungsmoduls konzipiert, die unabhängig von den betrachteten Montageprozessen Anwendung finden können und somit den Ebenen eins bis drei des Aufgabenmodells zugeordnet werden können. Dies betrifft z. B. die Identifikation von Sekundärprozessen. Diese sind unabhängig von den später betrachteten Prozessen und können immer verwendet werden.

5.4.1 Detaillierung der Anforderungen

In diesem Abschnitt werden die allgemein aufgestellten Anforderungen auf den Kern des aufgabenorientierten Programmiersystems, das Planungsmodul, heruntergebrochen. Von besonderer Bedeutung ist die *Erweiterbarkeit und Offenheit*, da zum einen bei der Implementierung des Systems kaum alle möglichen Montagesysteme und -prozesse berücksichtigt werden können und zum anderen existierende Software und Simulationssysteme integrierbar bzw. bestehende Softwarefunktionen einfach ersetzbar sein sollten. Die *Transparenz* des Systems ermöglicht es dem Entwickler die Erweiterbarkeit zu nutzen. Eine weitere grundlegende Anforderung ist ein *effizienter Problemlösungsprozess* der eine zielgerichtete Arbeitsweise, aber auch eine schnelle Reaktion auf Sackgassen, voraussetzt. Eine *flexible Kontrolle* des Lösungsprozesses ermöglicht ein Auslassen von Teilschritten. Hat der Nutzer spezifisches Wissen, kann diese ideal eingebunden werden. Damit wird eine Flexibilität in der Komplexität der Aufgabenbeschreibung ermöglicht. Das Planungsmodul muss in seiner Struktur auf dem vorgestellten Informationsmodell aufbauen, um die Spezifika von Ressourcen zu abstrahieren. Somit soll ein Skill-basierter Planungsprozess umgesetzt werden.

5.4.2 Auswahl eines Architekturmusters

Architekturmuster liefern Grundstrukturen bestehender Software-Systeme mit ähnlichen Problemen und tragen zur Transparenz des zu entwickelnden Systems bei (BUSCHMANN 2001). Sie beschreiben also den Stil einer Gesamtarchitektur eines Systems (HASSELBRING 2006). Beispiele hierfür sind die Schichtarchitektur, Pipes-and-Filters Architektur oder die Blackboard Architektur (BUSCHMANN 2001). Für die oben genannten Anforderungen ist ein geeignetes Architekturmuster auszuwählen, auf dessen Basis die Konzeption des Planungsmoduls erfolgen

kann. Bei einem Vergleich bestehender Architekturmuster wurden die Blackboard[3] Architektur sowie die Service Orientied Architecture (SOA) als besonders vielversprechend bewertet. Diese weisen u. a. bzgl. der Kapselung der Funktionalitäten starke Ähnlichkeiten auf. Für den Anwendungsfall dieser Arbeit wurde die Blackboard Architektur ausgewählt. Diese erfüllt alle Anforderung und wurde, im Gegensatz zur SOA, speziell für Problemlösungsprozesse konzipiert. So verwendet insbesondere HUMBURGER (1998) auch einen Blackboard-Ansatz bei der Konzeption seines aufgabenorientierten Programmiersystems, welches als Grundlage dienen kann.

Wie jedes wissensbasierte System besteht auch die Blackboard Architektur aus drei wesentlichen miteinander kommunizierenden Komponenten: Die *Experten* mit problemspezifischen Regeln, Algorithmen und Heuristiken, die *Steuerung* mit Kontrolldaten zum Dirigieren des Lösungsprozesses und die *globale Datenbasis (Blackboard)* mit allen problemspezifischen Fakten und Zwischenlösungen. Die Steuerung entspricht einem Leiter der in der Metapher benannten Experten, welcher ihre Beiträge koordiniert. Die Experten werden in diesem Zusammenhang auch Wissensquellen genannt (LEVI 1988). Zwei Planungsmethoden sind charakteristisch für den Lösungsprozess einer Blackboard Architektur. Zum einen die hierarchische Problemlösung, d. h. die globale Lösung wird als eine Hierarchie von lokalen Lösungen aufgebaut. Zum anderen die opportunistischen Lösungseinheiten, d. h. einzelne Lösungen können separat entwickelt werden und der Problemlösungsprozess wird auf diejenigen Lösungen fokussiert, die am erfolgversprechendsten sind (LEVI 1988).

Den Ablauf der Problemlösung beschreibt HUMBURGER (1998) in Anlehnung an NII (1986) wie folgt: Die Wissensquellen stellen Funktionen zur Problemlösung bereit und sind in einen Bedingungs- und Aktionsteil unterteilt. Ist der Bedingungsteil erfüllt, kann der Aktionsteil, in dem die spezielle Funktionalität implementiert ist, ausgeführt werden. Die Wissensquellen kommunizieren nur über das Blackboard miteinander. Jede Wissensquelle überwacht mit ihrem Bedingungsteil das Blackboard und meldet an die Steuerung wenn sie ausführbar ist. Die Steuerung bestimmt den Fokus der Aufmerksamkeit, sie entscheidet, welche Wissensquelle als nächstes ausgeführt wird. Die Wissensquelle modifiziert das Blackboard um ihren Lösungsbeitrag, durch den möglicherweise neue Wissensquellen ausführbar werden. Das Vorgehen wird so lange wiederholt bis keine

[3] „Der Begriff leitet sich von der Metapher einer Kreidetafel (engl. Blackboard) ab, um die eine Reihe von Experten sitzt, die gemeinsam ein Problem lösen müssen, indem sie die Tafel als Medium zur Kommunikation benutzen“ (ZÖLLER-GREER 2010, S. 127).

Wissensquelle mehr ausführbar ist. Eine detaillierte Beschreibung der Blackboard Architektur kann z. B. in QIAN ET AL. (2010) nachgelesen werden. Da die Blackboard Architektur in ihrem Grundaufbau nichtdeterministisch ist (ZÖLLER-GREER 2010, S. 127), muss in der Konzeption besonderer Wert auf die Steuerung gelegt werden, um dieses mögliche Problem zu umgehen. Der klar strukturierbare Ablauf bei der Montageplanung liefert hier jedoch eine gute Grundlage.

5.4.3 Überblick der Architektur des Planungsmoduls

5.4.3.1 Entwurfsentscheidungen

Steuerung

Dem Aufbau der Steuerung bzw. Kontrolle kommt eine besondere Bedeutung zu, da diese den Planungsablauf koordiniert. Der Aufbau beeinflusst also direkt die Effizienz des Problemlösungsprozesses und dessen Transparenz. Zwei mögliche Ansätze stellen HUMBURGER (1998) und GANGHOFF (1993) vor. HUMBURGER (1998) realisiert die Steuerungskomponente als Meta-Ebenen-Architektur mit einem wissensquellenorientierten Steuerungskonzept, d. h. dass bei Aufruf der Wissensquellen der Fokus der Aufmerksamkeit auf den Triggerbedingungen der Wissensquellen liegt und diese bei Erfüllung der Bedingungen aufgerufen werden können. Das Konzept wurde in Kapitel 3 näher beschrieben. GANGHOFF (1993) konzipiert in seinem wissensbasierten Planungswerkzeug einen hybriden Kontrollansatz aus vorgehens- und merkmalsorientierter Kontrolle. Mit der vorgehensorientierten Kontrolle soll die effiziente Verwendung von domänenspezifischen Problemlösungswissen, wie z. B. Bearbeitungsreihenfolgen oder Vorgehensalternativen ermöglicht werden. Das Lösungswissen wird in Form von UND/ODER-Bäumen aufbereitet, was einen automatischen Planungsprozess ermöglicht. Sie eignet sich für die Um- und Änderungsplanung.

Grundsätzlich sind beide Ansätze zur Weiterverwendung im zu konzipierenden erweiterten Planungsmodul geeignet. Der hierarchische Aufbau der Meta-Ebenen-Architektur mit dem wissensquellenorientierten Ansatz ist jedoch transparenter und die Kapselung des Wissens auf verschiedenen Ebenen erleichtert die Erweiterung und die Anpassung. Für den strukturellen Aufbau der Steuerung wird daher der Ansatz nach HUMBURGER (1998) verwendet. Der Ablauf und die Inhalte der Steuerung werden neu entworfen.

Optimierung im Planungsprozess

HUMBURGER (1998) definiert zwei grundsätzliche Ansatzpunkte der Optimierung für die aufgabenorientierte Programmierung. Dies ist zum einen die lokale Optimierung für Teilergebnisse, wie z. B. der Planung einer Transferbewegung. Zum anderen ist dies die globale Optimierung, welche die lokale Optimierung anhand von Gütekriterien steuert. Die Optimierungsaufgabe kann als Graph beschrieben werden, der aber nicht a priori bekannt ist. Dieser universelle Ansatz soll auch für diese Arbeit Anwendung finden.

Struktur des Blackboards und Hierarchie des Problemlösungsprozesses

Das in Unterkapitel 5.3 vorgestellte Aufgaben- sowie Umweltmodell stellen die wesentlichen Bestandteile des Blackboards dar und werden an dieser Stelle nicht nochmals beschrieben. Der Problemlösungsprozess orientiert sich an den Ebenen des Aufgabenmodells, d. h. die Planung erfolgt von der Primärprozessebene, Montagesquenzebene, Sekundärprozessebene und Skillebene bis zur Parameterebene. Die Optimierung erfolgt nur über die drei zuletzt genannten Ebenen.

5.4.3.2 Architektur des Planungsmoduls

Abbildung 35 zeigt die entwickelte Architektur des Planungsmoduls. Die Ebenen der Architektur orientieren sich an dem von HUMBURGER (1998) vorgeschlagenen Konzept, die Inhalte und auch der Aufbau des Blackboards wurde jedoch neu entwickelt.

In der Strategieebene ist die Strategiewissensquelle für die Leitung des Planungsablaufs zuständig. Der Problemlösungsvorgang wird mittels des Blackboards und durch Rückmeldungen aus niedrigeren Hierarchieebenen überwacht, um daraufhin die auszuführenden Aufgaben zu verteilen. Die Strategie ruft sowohl die Wissensquellen in der Spezialisten- als auch in der Aufgabenebene auf und koordiniert somit den gesamten Planungsablauf.

Die Aufgabenebene ist horizontal in Entscheidungs- und Planungswissen gegliedert und repräsentiert Planungsaufgaben, die unabhängig von der Planung einzelner Prozesse sind. Die in dieser Ebene enthaltenen Funktionen werden im Folgenden auch als Kontrollwissensquellen bezeichnet. Das Planungswissen ist nochmal anhand der definierten Hierarchieebenen aufgeteilt. Jede Ebene repräsentiert eine spezielle Planungsaufgabe mit dem dazu nötigen Planungswissen. Der Primärprozessebene ist eine Funktion zur Ableitung der Reihenfolge der Primärprozesse aus den Produktdaten zugeordnet. Funktionen der Sekundärpro-

zessebene sind „Ressourcenzuordnung Primärprozesse“, „Identifikation Sekundärprozesse“ und „Ressourcenzuordnung Sekundärprozesse“.

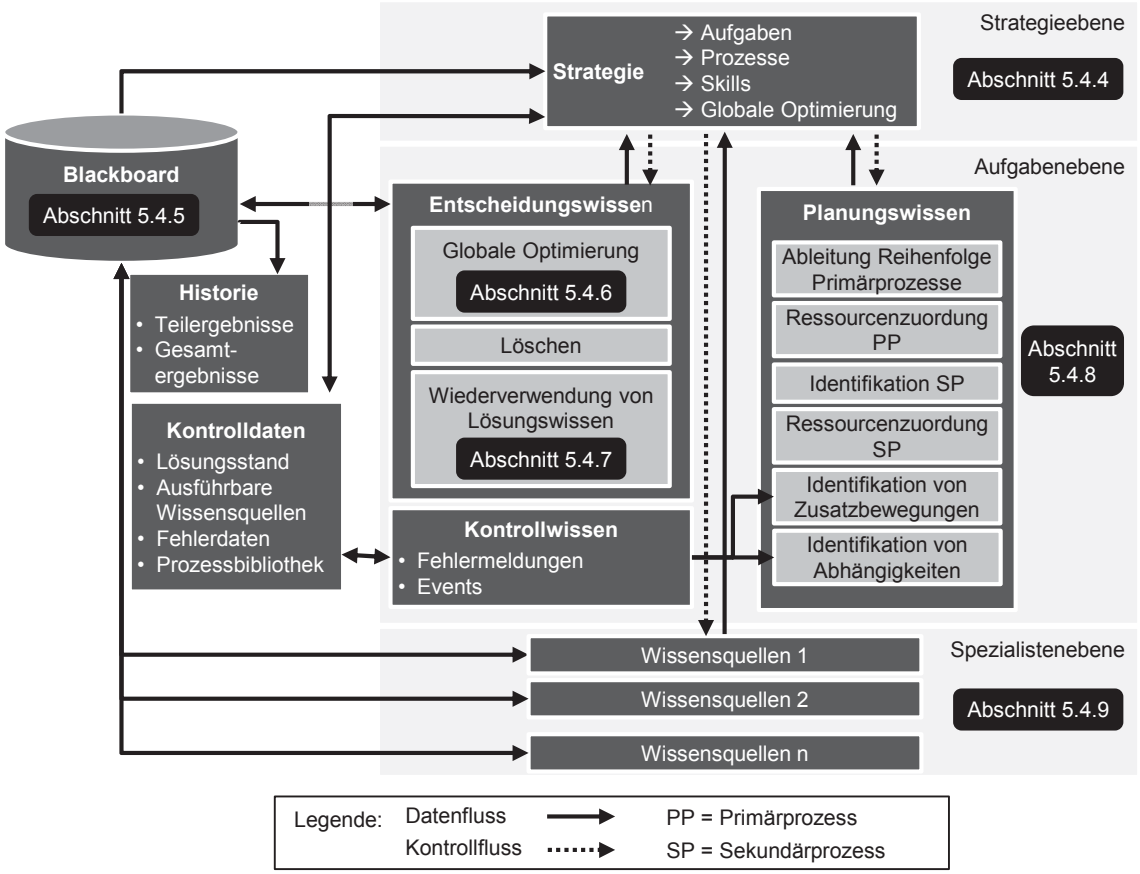


Abbildung 35: Architektur des Planungsmoduls in Anlehnung an HUMBURGER (1998)

Die Planungswissensquellen „Identifikation von Zusatzbewegungen“ und „Identifikation von Abhängigkeiten“ sind Teil der Skillebene und können auf gesondert gesammeltes Kontrollwissen zugreifen. Dieses Kontrollwissen beinhaltet bestimmte Ereignisse oder Fehler, die in der Planerstellung auftreten und für die es vordefinierte Lösungswege gibt. Die beiden zuvor genannten Wissensquellen greifen diese Ereignisse oder Fehler auf, um dafür Lösungen zu generieren. Das Entscheidungswissen stellt Funktionen zur Entscheidungsfindung und -durchsetzung bereit. Die Entscheidungswissensquelle „Optimierung“ soll in einer Auswahl von alternativen Lösungswegen den Pfad finden, der die beste Lösung nach einem definierten Kriterium (z. B. Durchlaufzeit, Energieverbrauch, …) darstellt. Die Entscheidungswissensquelle „Löschen“ stellt Methoden zum Löschen einzelner Lösungsalternativen oder ganzer Lösungspfade zur Verfügung. Die Entscheidungswissensquelle „Wiederverwendung von Lösungswissen“ soll Funktionen zur Wiederverwendung des durch abgeschlossene Planungen erlang-

ten Wissens (Historie) bereitstellen. Sie erlaubt die Wiederverwendung gesamter abgeschlossener Planungen oder das Heranziehen von Teilausschnitten vergangener Planungen zur Neuerstellung von Montageplänen. Speziell bei der Produktion von verschiedenen Varianten eines Produkts muss nicht die vollständige Planung durchgeführt werden, lediglich die abweichenden Planungsschritte bedürften einer Neuplanung. Dadurch kann die Effizienz im Lösungsprozess weiter gesteigert werden.

Die Spezialistenebene enthält das durch die Wissensquellen repräsentierte spezifische Domänenwissen. Die Wissensquellen beinhalten Bearbeitungs- und Reihenfolgenwissen zur Durchführung der Montageaufgaben und können in Wissensquellen für Primär- (Fügeplaner) und Sekundärprozesse (Handhabungs- und Sensorplaner) eingeteilt werden. Sie stellen die Algorithmen zur Planung bereit und verfügen jeweils über eigene Datenbanken. Beispiele sind die Wissensquelle zur Planung eines robotergeführten Klebeprozesses oder die Handhabungsplanung. Sie besitzten vordefinierte Schnittstellen und müssen die notwendigen Parameter für die globale Optimierung bereitstellen. In dieser Ebene sind entsprechend auch typischerweise Simulationssysteme angebunden.

Die Kontrolldaten enthalten alle für den Planungsprozess nötigen zusätzlichen Informationen. Dazu gehören der aktuelle Lösungsfortschritt, die ausführbaren Wissensquellen und benötigte Daten zur erfolgreichen Fehlerbehebung und Identifikation von Abhängigkeiten. Sie umfassen auch eine Bibliothek der dem Planungsmodul bekannten Prozesse, die insbesondere beim Einfügen von Sekundärprozessen zur Anwendung kommen.

Entsprechend dem Architekturmuster dient das Blackboard als Datenbasis für den Problemlösungsprozess. In ihm erfolgt die Problemlösung, indem durch die einzelnen Wissensquellen Änderungen vorgenommen werden. Der Aufbau des Blackboards und der Problemlösungsprozess werden nachfolgend beschrieben.

5.4.4 Koordination des Planungsprozesses durch die Strategie

Die Strategiewissensquelle ist auf der obersten Ebene der Architektur angesiedelt. Sie koordiniert die Planerstellung und entscheidet, welche Aktion als nächstes ausgeführt werden soll. Ihre Entscheidungen trifft sie auf Basis des Blackboards, der Kontrolldaten und den Rückmeldungen aus Wissensquellen niedrigerer Ebenen (HUMBURGER 1998). Im Folgenden wird zuerst das grundsätzliche Vorgehen der Strategiewissensquelle erklärt. Anschließend wird der Aufbau der Kontrolldaten dargestellt und darauf eingegangen, wer diese erstellt und verwen-

det. Der durch die Strategiewissensquelle gesteuerte Planungsablauf kann in drei Abschnitte unterteilt werden (vgl. Abbildung 36).

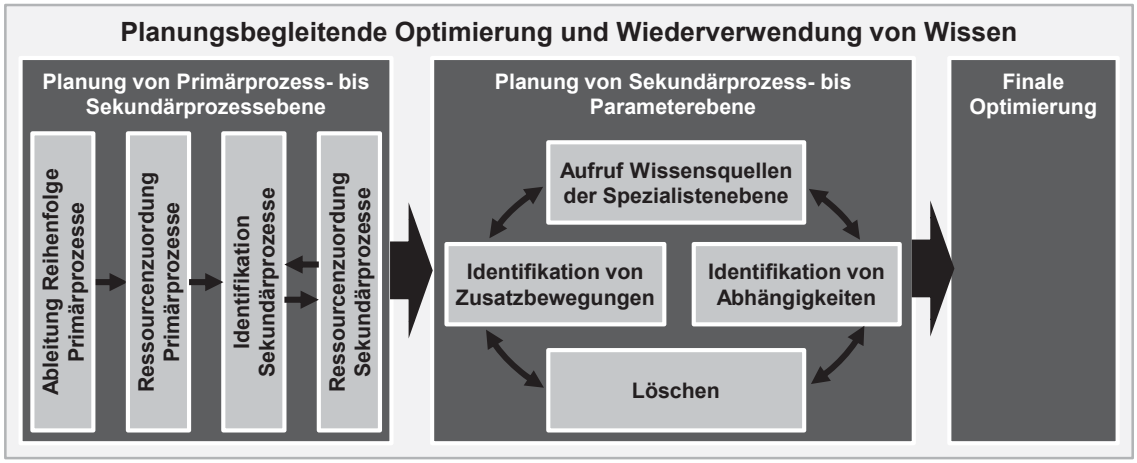


Abbildung 36: Durch die Strategie gesteuerter Ablauf der Planung

Zu Beginn der Planung steht das durch den Nutzer definierte Aufgabenmodell. In einem ersten Schritt ruft die Strategiewissensquelle der Reihe nach die Planungswissensquellen „Ableitung Reihenfolge Primärprozesse“, „Ressourcenzuordnung Primärprozesse“, „Identifikation Sekundärprozesse“ und „Ressourcenzuordnung Sekundärprozesse“ auf. Die beiden letztgenannten Planungswissensquellen können auch mehrfach aufgerufen werden, wenn sich aus der Zuordnung der Ressourcen der Sekundärprozesse ergibt, dass z. B. noch weitere Handhabungsprozesse notwendig sind. Ein Abbruchkriterium verhindert, dass weitere Sekundärprozesse eingefügt werden, obwohl keine Ressourcen zugeordnet werden können. Die Planungswissensquellen geben nach erfolgreicher Bearbeitung eine Rückmeldung an die Strategie und vermerken dies auch in den Kontrolldaten. Aus den Daten des Blackboards kann die Strategie schließen, wenn Schritte übersprungen werden können. Dies ist z. B. der Fall, wenn die Aufgabe auf Ebene der Sekundärprozesse beschrieben wurde. Im zweiten Abschnitt erfolgt die Detaillierung der Prozess-Steps durch die Wissensquellen der Spezialistenebene. Die Strategie koordiniert hierbei den Aufruf der Wissensquellen der Spezialistenebene im Wechsel mit den Planungswissensquellen „Identifikation von Zusatzbewegungen“, „Identifikation von Abhängigkeiten“ und „Löschen“. Die Wissensquellen werden solange aufgerufen, bis der SFC vollständig durch Skills mit deren zugehörigen Parametern ausgeplant ist. Die Optimierungsfunktion kann variabel während des kompletten Planungsprozesses eingesetzt werden. Sie kann zu jeder Zeit auf das komplette SFC oder einen ausgewählten Abschnitt angewendet werden. Dadurch können bereits während des Planungsprozesses

iterative Entscheidungen hin zur optimalen Lösung getroffen werden. Mögliche aber nicht sinnvolle Lösungswege können frühzeitig verworfen werden. In jedem Fall findet jedoch nach Abschluss des Planungsprozesses die Optimierung statt.

Die Steuerungsstrategie im zweiten Abschnitt zur Ausplanung der Prozesse durch Skills wird nachstehend genauer betrachtet. Die Wissensquellen der Spezialistenebene beobachten selbstständig die Veränderungen auf dem Blackboard und melden wenn sie ausführbar sind. Sie werden dann als ausführbar in die Kontrolldaten eingetragen. Die Strategiewissensquelle wählt die dort notierten Wissensquellen aus und ruft sie auf. In der Planung haben Primärprozesse Priorität vor Sekundärprozessen. Die Wissensquellen definieren dabei auch die Positionen zwischen den einzelnen Skills. Die Planungswissensquellen „Identifikation von Zusatzbewegungen“ und „Identifikation von Abhängigkeiten“ werden nach Rückmeldung aus den Wissensquellen der Spezialistenebene je nach Bedarf von der Strategiewissensquelle aufgerufen. Fehlerdaten und Übergangsinfos werden in den Kontrolldaten gespeichert. Die Reihenfolge, in der die Wissensquellen der Spezialistenebene aufgerufen werden, wird gespeichert. Durch die Modellierung der Skills mit zugehörigen Zustandsdiagrammen, kann die Strategie darüber hinaus automatisiert überprüfen, ob alle Skills wieder in den Ausgangszustand zurückversetzt worden sind. Ein Beispiel ist eine Greifer, der nach dem schließen am Ende des Prozesses auch wieder geöffnet werden sollet. Schlägt die Planung fehl und eine Neuplanung ist erforderlich, können beim zweiten Planungsdurchlauf die Wissensquellen in einer veränderten Reihenfolge aufgerufen werden. Die Reihenfolge wird durch die Strategie festgelegt. Zudem kann die Prioritätsregel, bzgl. Primär- und Sekundärprozessen außer Kraft gesetzt werden, um Planungsfehler nicht zu wiederholen. Am Ende des zweiten Abschnitts, wenn keine Wissensquelle mehr ausführbar ist, überprüft die Strategiewissensquelle, ob alle Prozesse ausgeplant wurden und notiert dies in den Kontrolldaten. Andernfalls ist die Planung vorerst fehlgeschlagen und es wird analysiert, wieso nicht alle Prozesse ausgeplant werden konnten. Ist die Ursache auf fehlende Wissensquellen zurückzuführen, muss die Planung abgebrochen werden. Ansonsten kann entweder ein neuer Planungsversuch mit einer veränderten Reihenfolge der Aufrufe der Wissensquellen oder nach einer Überarbeitung der Eingabe durch den Nutzer eine neue Planung gestartet werden.

In den Kontrolldaten werden folgende Ereignisse vermerkt: Die Vollständigkeit der Nutzereingaben, die erfolgte Ressourcenzuordnung für Primär- und Sekundär-prozesse, die erfolgte Identifikation der Sekundärprozesse, den Abschluss der Planung, d. h. dass alle Prozesse ausgeplant wurden, ausführbaren Wissensquel-

len, Fehlerdaten bzw. fehlgeschlagenen Planungen, Übergabeevents, die Reihenfolge der aufgerufenen Wissensquellen sowie den Erfolg der Optimierung.

5.4.5 Problemlösung im Blackboard

Im Blackboard werden die Inhalte des Informationsmodells, d. h. das Umwelt- und Aufgabenmodell instanziiert. Daher ist das Blackboard auch als AutomationML-Datei mit den in Abschnitt 5.3.8 beschriebenen Teilformaten aufgebaut. Da das Aufgabenmodell keine fest definierte Form, sondern mehrere mögliche Ebenen darstellt, ist auch der detaillierte Inhalt des Blackboards dynamisch. Das Aufgabenmodell und dessen Ebenen stellen wie beschrieben auch die Ebenen im Problemlösungsprozess dar. Abbildung 37 zeigt den Aufbau des Blackboards. Da Primär-, Sekundär- und Skillebene während des Planungsprozesses somit ineinander übergehen, dient eine einzige Sequenz, integriert in einen PLCopen XML Sequential Function Chart (SFC), als dynamisches Element für den Planungsprozess. Statische Bestandteile wie Produkte, Prozesse mit zugehörigen Skills oder alleinstehende Skills auf Skillsebene, werden im SFC referenziert. Deren Reihenfolge und Instanziierung im SFC ist somit auch dynamisch.

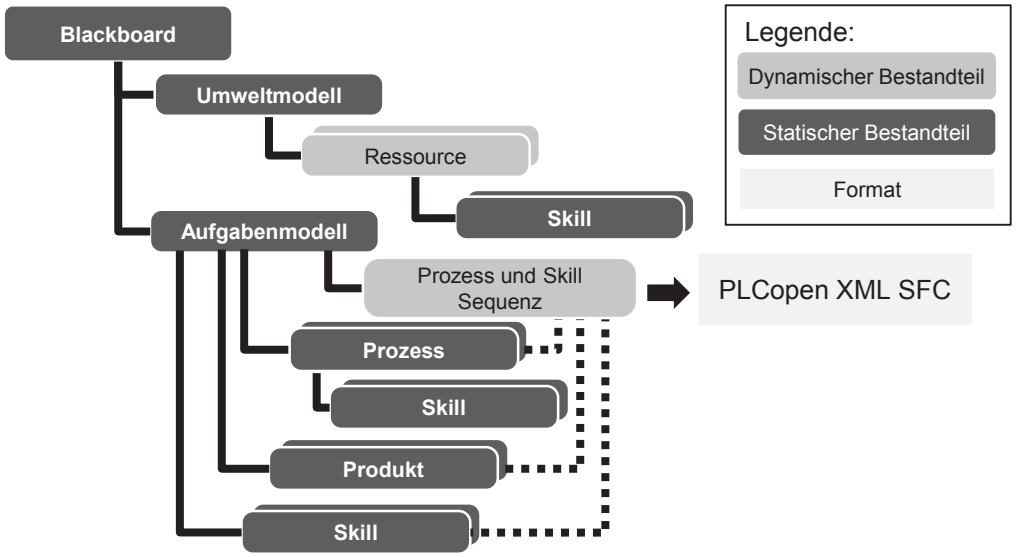


Abbildung 37: Schematischer Aufbau des Blackboards

In einem SFC können durch Schritte, Transitionen, Divergenzen und Konvergenzen parallele und sequentielle Abläufe dargestellt werden. Für die Ablaufmodellierung mit Prozessen und Skills werden spezielle Prozess- und Skill-Schritte vordefiniert, die mit den Instanzen in AutomationML referenziert sind. Die Transitionen zwischen den Schritten definieren die Übergangsbedingungen. An diesen ist der jeweilige Zustand des Produkts vor bzw. nach der durch den Prozess-

bzw. Skill-Schritten durchgeführten Änderung beschrieben, indem diese dort jeweils mit addData-Elementen[4] referenziert werden.

Während der Lösungsfindung erfolgt zunächst die Ableitung der Primärprozesssequenz, falls diese nicht gegeben ist. Anschließend wird der SFC mit den notwendigen Sekundärprozessen erweitert. Im nächsten Schritt planen und detaillieren die Wissensquellen der Spezialistenebene die Prozesse. Während der Bearbeitung durch die Wissensquellen werden die Prozess-Schritte nacheinander durch die Skill-Schritte ausgeplant, bis am Ende ein detaillierter Ablauf, beschrieben durch Skills, entstanden ist. Abbildung 38 zeigt die Detaillierung der Prozess-Schritte durch Skill-Schritte am Beispiel des Handhabens.

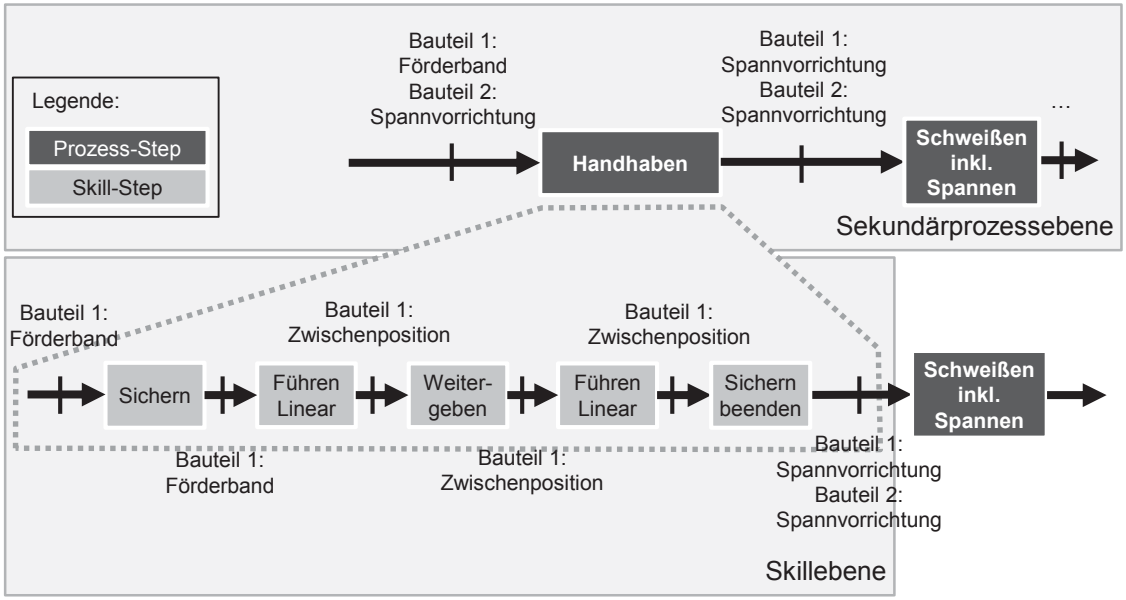


Abbildung 38: Detaillierung von Prozess-Steps zu Skill-Steps im Problemlösungsprozess

5.4.6 Optimierung des Ablaufs im Rahmen des Planungsprozesses

Wie beschrieben, erfolgt die Optimierung in zwei Ebenen. Die Wissensquellen auf Spezialistenebene sind dafür zuständig, dass diese ein nach vorgegebenen Kriterien (z. B. Taktzeit oder Energieverbrauch) optimales Ergebnis in den SFC schreiben. Dies umfasst jedoch nicht die Vorauswahl der möglichen Alternativen, die sich z. B. aus verschiedenen Aufspannungen ergeben. Diese werden als Alternativen in den SFC geschrieben, enthalten aber jeweils die optimale Bahn.

[4] Mit den addData-Elementen können anwenderspezifische Informationen als zusätzliche Elemente an den verschiedenen Objekten in PLCopen XML hinterlegt werden. Der Inhalt der verwendeten addData-Elemente wird am Dokumentanfang über die addData-Info spezifiziert und kann frei definiert werden (DRATH, S. 157).

Auf übergeordneter Ebene sind während und am Ende des Planungsprozesses die Entscheidungen zwischen Ablaufalternativen im Lösungsprozess zu treffen. Dies erfolgt basierend auf Kriterien wie Durchlaufzeit oder Energieverbrauch. Im Folgenden werden die Auswahl eines Optimierungskonzepts und dessen Aufbau beschrieben.

5.4.6.1 Auswahl einer Optimierungsstrategie zur Entscheidung zwischen Alternativen

Für die Auswahl von Alternativen können verschiedene Ansätze zur Anwendung kommen. In Bereich der Produktionstechnik und Montage sind Ansätze wie Graphensuchverfahren (LAU 2010), Kostenfunktion bzw. Kostenwerte (HUMBURGER 1998; BÜSCHER ET AL. 2013) oder genetische Algorithmen (KASHKOUSH & ELMARAGHY 2014) verbreitet. Mit dem Fokus der Auswahl von Alternativen sind insbesondere Graphensuchverfahren und Kostenfunktionen geeignet. Im Vergleich zur Kostenfunktion bietet das Graphensuchverfahren jedoch die nachstehend genannten Vorteile und wird daher in dieser Arbeit verwendet. Wird das System um neue Module bzw. Wissensquellen erweitert, müssen diese lediglich über eine Funktion zur Berechnung des für die Optimierung nötigen Werts verfügen. Es ist keine spezielle Einbindung dieser Module in den Optimierungsalgorithmus notwendig. Bei einer Optimierung mit einer Kostenfunktion müsste das Kostenmodell angepasst werden.

5.4.6.2 Algorithmus und Ablauf der Optimierung

Das Graphensuchverfahren, das in der Optimierung Anwendung finden soll, beschreibt die Lösung des kürzesten-Pfad-Problems in einem Graphen. Für die Lösung des kürzesten-Pfad-Problems eignen sich nach CORMEN ET AL. (2013) folgende Algorithmen: Bellman-Ford-Algorithmus, A*-Algorithmus und-Dijkstra-Algorithmus. Für eine detaillierte Beschreibung der Algorithmen wird an dieser Stelle auf weiterführende Literatur verwiesen (SATTLER & SAAKE 2010; CORMEN ET AL. 2013). Wegen der geringen Laufzeit für die Umsetzung im Vergleich zum Bellman-Ford-Algorithmus (CORMEN ET AL. 2013) wurde der Dijkstra-Algorithmus ausgewählt. Vor allem bei komplexen Montagesystemen kann sich einer sehr große Anzahl an Alternativen ergeben, aus welchen effizient eine Lösung ausgewählt werden muss. Der Dijkstra-Algorithmus findet in einem gewichteten, gerichteten Graph den kürzesten Pfad zwischen einem Startknoten

und allen anderen Knoten des Graphen unter der Voraussetzung, dass alle Kantengewichte positiv sind (CORMEN ET AL. 2013).

Die Optimierungsfunktion kann von der Strategiewissensquelle variabel im Lösungsprozess aufgerufen werden, um zwischen alternativen Lösungspfaden zu entscheiden. Dabei kann sie einen Start- und Zielknoten vorgeben, um einen Ausschnitt des SFCs zu optimieren, oder sie kann die Optimierung auf das komplette SFC anwenden. Abbildung 39 zeigt den Ablauf der Optimierung. In einem ersten Schritt ist es notwendig, den SFC in einen Graphen zu transformieren. Zwar entspricht der SFC weitestgehend einem Graphen, jedoch müssen bei der Umwandlung folgende Aspekte beachtet werden. Im SFC werden alle Alternativ- und Parallelverzweigungen und -zusammenführungen mit Nummern versehen. Diese Stellen entsprechen den Knotenpunkten im Graphen. Im SFC besitzen nur die Schritte Gewichte, da dort die Bearbeitungsschritte stattfinden bzw. die Optimierungsgrößen hinterlegt sind. Die Gewichte aufeinanderfolgender Schritte wird aufsummiert und bei parallelen Bearbeitungspfaden wird der Pfad mit dem höchsten Gewicht ausgewählt sowie im Graph eingetragen. Alle Transformationsschritte werden dokumentiert, um eine erfolgreiche Rücktransformation nach Durchführung der Optimierung und Auswahl des optimalen Pfads sicherzustellen.

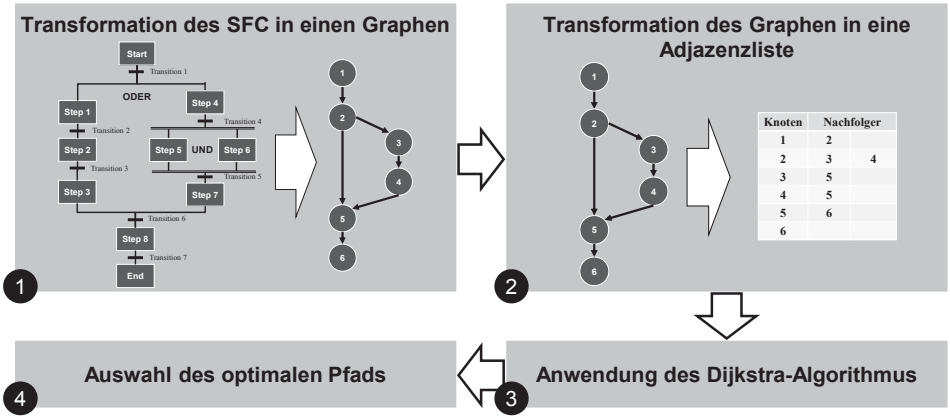


Abbildung 39: Ablauf der Optimierung des Ablaufs

Im zweiten Schritt wird der Graph in eine Adjazenzliste überführt. Eine Adjazenzliste enthält für jeden Knoten im Graph eine eigene Liste. Die Liste eines Knotens enthält alle Nachfolgeknoten mit denen der Knoten direkt über eine ausgehende Kante verbunden ist. Die Gewichte der ausgehenden Kanten werden gemeinsam mit dem Nachfolgeknoten in der Liste vermerkt (CORMEN ET AL. 2013). Die Durchführung des Dijkstra-Algorithmus im dritten Schritt kann bei CORMEN ET AL. (2013, S. 671) an einem Beispiel nachgelesen werden. Dessen

genauer Ablauf wird daher an dieser Stelle nicht dargestellt. Mit dem Ergebnis des Algorithmus kann im letzten Schritt über die Dokumentation der Transformationsschritte die Auswahl des optimalen Pfads erfolgen.

5.4.7 Wiederverwendung von bestehendem Lösungswissen

Um bestehende Lösungen und Teillösungen wiederverwenden zu können, ist in der Architektur eine eigenständige Wissensquelle vorgesehen. Diese überwacht zyklisch den Problemlösungsprozess und greift, wenn bekannte Zustände im SFC detektiert werden, auf die Historie zurück. Als Algorithmus zur Umsetzung der beschriebenen Funktionalität eignen sich insbesondere die von BÜSCHER ET AL. (2013) beschriebene kognitive SOAR (State, Operator Apply Result) Architektur oder genetische Algorithmen wie sie von KASHKOUSH & ELMARAGHY (2014) vorgeschlagen werden. Wegen der geringeren Anzahl an notwendigen Daten in der Historie erscheint somit insbesondere der erste Ansatz als zielführend. Für die Ausgestaltung der Funktionalität sei daher auf diese Arbeiten verwiesen.

5.4.8 Übergreifende Wissensquellen in der Aufgabenebene

Die Planungswissensquellen übernehmen die Planung in der Aufgaben-, Primär- und Sekundärprozessebene mit der Ableitung der Primärprozesssequenz, der Ressourcenzuordnung und der Identifikation der Sekundärprozesse. In der Skillebene unterstützen sie die Planung der Wissensquellen der Spezialistenebene durch eine Möglichkeit der Fehlerbehebung und Plananpassung und durch die Koordination von Überschneidungen und Abhängigkeiten zwischen der verwendeten Skills. Da diese Wissensquellen übergreifend, d. h. unabhängig von den geplanten Montageprozessen sind, wurden deren Funktionalitäten entwickelt. Sie bilden die Basis, um das Programmiersystem an beliebige Montageprozesse anzupassen. Dies erfolgt mit den Wissensquellen der Spezialistenebene.

5.4.8.1 Ableitung der Reihenfolge von Primärprozessen

Für die Ableitung einer Montagereihenfolge aus Produktdaten wurde in der Vergangenheit in einer Vielzahl von Konzepten erarbeitet (KAUFMAN ET AL. 1996; ENG ET AL. 1999; MOSEMANN 2000; THOMAS 2008). Typischerweise wird hier ein Assembly-by-Disassembly-Ansatz verfolgt, mit dem ausgehend von den geometrischen Randbedingungen der Bestandteile eines vollständig montierten Produktes über räumliche Operationen und Kollisionsüberprüfung die Montagereihenfolge identifiziert werden kann. Wegen der bestehenden Ansätze wird kein

neues Konzept entwickelt, sondern insbesondere auf die Arbeiten von THOMAS (2008) und KAUFMAN ET AL. (1996) verwiesen.

5.4.8.2 Ressourcenzuordnung für Primär- und Sekundärprozesse

Die Ressourcenzuordnung identifiziert mögliche Ressourcen für Prozesse und ist in die Ressourcenzuordnung für Primär- und Sekundärprozesse unterteilt. Der grundsätzliche Ablauf der Ressourcenzuordnung ist für Primär- und Sekundärprozesse jedoch nahezu gleich. Die in den Primär- und Sekundärprozessen definierten Skills können mit den instanziierten Skills der im Umweltmodell beschriebenen Ressourcen abgeglichen werden. Der Abgleich erfolgt über die ID des Skills aus dem Blackboard. Für einzelne Skills, wie z. B. der Traglast für eine Bewegung, kann an dieser Stelle auch ein grober Abgleich der Eigenschaften der Skills und der Vorgaben aus der Produkt und Prozessbeschreibung erfolgen. Es sind dazu jedoch für jeden Skill spezielle Funktionalitäten für deren Abgleich notwendig. Ein Konzept hierfür wird z. B. von MICHNIEWICZ & REINHART (2014) beschrieben. Der detaillierte Abgleich der Eigenschaften der Skills einer Ressource mit den Anforderungen des Produktes und Prozesses erfolgt erst durch die Wissensquellen auf Spezialistenebene. Dies ist darin begründet, dass für viele Prozesse zunächst eine detailliertere Planung mit den Prozessvorgaben notwendig ist, um eine Überprüfung zu ermöglichen. Als Beispiel sei an dieser Stelle ein Schweißprozess genannt. Bei diesem müssen zunächst abhängig vom Werkstoff und der Nahtgeometrie die Schweißparameter wie Geschwindigkeit und Spannung abgeleitet werden. Erst mit diesen ist es möglich, zu überprüfen, ob z. B. die erreichbaren Geschwindigkeiten der Ressource ausreichen. Sind mehrere Ressourcen für die Bearbeitung eines Prozesses möglich, so werden alternative Verzweigungen in den SFC eingefügt.

Bei der Ressourcenzuordnung der Primärprozesse erfolgt noch keine Überprüfung der Vorbedingungen für den Prozess. D. h. in diesem Schritt wird z. B. noch nicht durch einen Abgleich der Arbeitsbereiche der Ressourcen und der Frames der zu montierenden Produkte überprüft, ob die Produkte schon den notwendigen Frame für die Bearbeitung haben. Für die Zuordnung von Sekundärprozessen zur Handhabung wird überprüft, ob Anfangs- und Endframe der Produkte im Arbeitsbereich der Ressource liegen. Ist dies nur für einen Frame der Fall, so wird zunächst die passende Ressource zugeordnet und der Arbeitsraum der Ressource als Frame des Produktes vor dem Prozess zugeordnet. Durch die Strategie wird über die Wissensquelle *Identifikation Sekundärprozesse* ein weiterer Prozess eingefügt. Wird anschließend wieder überprüft, ob Ressourcen die geforderten

Bedingungen erfüllen können, werden also Überschneidungen der Arbeitsbereiche von Ressourcen gesucht. Dieser Vorgang kann bei mehreren notwendigen Übergaben auch zyklisch wiederholt werden. Durch dieses Vorgehen der Zuordnung und der Überprüfung, ob die notwendigen Positionen erreicht sind, werden also die möglichen Pfade des Produktes durch das Montagesystem identifiziert und als Alternativen für die weitere Planung gespeichert.

5.4.8.3 Identifikation der notwendigen Sekundärprozesse

Wenn durch den Anwender nur die Primärprozesse vorgegeben sind, d. h. die Beschreibung der Aufgabe unabhängig vom Montagesystem erfolgt, müssen die notwendigen Sekundärprozesse durch das Planungsmodul ermittelt werden. Als Eingangsgröße dient der SFC inklusive der definierten Primärprozesse mit den in der Ressourcenzuordnung gefundenen Ressourcen. An den Transitionen vor und nach den Prozessen sind die Produkte bzw. Produktzustände mit ihrem aktuellen Frame hinterlegt. Grundsätzlich können drei Arten von Sekundärprozessen bei der Identifikation unterschieden werden. Mit den Positionen der Produkte vor und nach dem Prozess kann unter Verwendung der Arbeitsbereiche der Ressourcen geschlussfolgert werden, ob zusätzliche Handhabungsprozesse notwendig sind. Dieser Vorgang ist zyklisch nötig sein, falls die Endposition nicht durch die Handhabung mit einer Ressource erreicht werden kann. Ist in der Produktbeschreibung der Frame nicht exakt definiert, so sind Prüf- bzw. Messprozesse mit der Verwendung von Sensorik einzufügen. Über die angegebene Genauigkeit des Frames des Produkts und der Vorgaben in den Primärprozessen kann darüber hinaus geschlossen werden, ob Justageprozesse notwendig sind. Dabei können Alternativen in der Reihenfolge der Sekundärprozesse entstehen. Diese werden durch die Planungswissensquelle identifiziert und als Verzweigungen im SFC generiert. Für die Planungswissensquelle *Identifikation der Sekundärprozesse* wird die Annahme getroffen, dass sich das Bauteil lediglich im Arbeitsbereich der Ressource befinden muss. Die exakte Position und Orientierung der Bauteile wird nicht betrachtet. Wird im späteren Verlauf durch eine Wissensquelle der Spezialistenebene festgestellt, dass z. B. um ein Bauteil schweißen zu können, das Bauteil nochmals um 180° gedreht werden muss, kann der dazu fehlende Handhabungsprozess durch die Planungswissensquelle *Identifikation von Zusatzbewegungen* an dieser Stelle nachträglich ergänzt werden.

5.4.8.4 Identifikation von Zusatzbewegungen

Während der Ausplanung der Prozesse durch Wissensquellen der Spezialistenebene können Probleme entstehen, die durch Bauteile oder ein Hindernis im Arbeitsbereich hervorgerufen werden. Es handelt sich hier um Fehler, die durch das nachträgliche Einfügen von Sekundärprozessen oder Hilfsbewegungen behoben werden können. Die Planungswissensquelle *Identifikation von Zusatzbewegungen* enthält Wissen zur Lösung solcher Probleme. In ihr sind bestimmte Fehler bzw. Fehlerevents mit einer Lösung vordefiniert. Ein Beispiel dazu: Der Handhabungsroboter, der die Bauteile in den Arbeitsbereich eines Schweißroboters gebracht hat, ist noch im Arbeitsbereich des Schweißroboters und steht diesem somit im Weg. Es kann keine kollisionsfreie Bahn für den Schweißprozess gefunden werden. Dies wird an die Strategieebene gemeldet, welche wiederum die Planungswissensquelle *Identifikation von Zusatzbewegunge*n aufruft, die mit der vordefinierten Problemlösung startet. Typische Fehlerevents bzw. -meldungen sind Kollisionen mit Gegenständen, die daraufhin gehandhabt werden müssen und Kollisionen mit beweglichen Ressourcen, welche dann mit einer Hilfsbewegung verfahren werden müssen. Die Planung der Bewegungen erfolgt in der Spezialistenebene.

5.4.8.5 Identifikation von Abhängigkeiten

Bei der detaillierten Ausplanung der Prozesse können Abhängigkeiten und Überschneidungen bei Handhabungsprozessen entstehen. Beispielsweise darf das Lösen und Wegfahren eines Handhabungsroboters mit Greifer erst erfolgen, wenn das Produkt in der Spannvorrichtung für den Schweißprozess fixiert ist. Sind für die Planung der beiden Prozesse unterschiedliche Wissensquellen auf der Spezialistenebene vorhanden, ist deren Abstimmung notwendig. Auf Basis der Reihenfolge der Prozessschritte können Ereignisse in den Protokolldaten abgelegt werden. Dazu muss detektiert werden, wann die Skills für das „Sichern“ und „Sichern beenden“ von Bauteilen aufeinanderfolgen. Ist dies der Fall, wird das Ereignis „Übergabe“ in den Kontrolldaten aktiviert und ein Platzhalter in den SFC eingefügt. Die nachfolgend aufgerufene Handhabungsplanung auf der Spezialistenebene erhält diese Übergabeinformationen und kann diese bei der Planung berücksichtigen und die Platzhalter ersetzen. Eine weitere Abhängigkeit kann sich ergeben, falls Ressourcen zur gleichen Zeit für Prozesse eingeplant werden. Beispielsweise wäre es möglich einen Handhabungsroboter gleichzeitig für verschiedene Handhabungsvorgänge einzuplanen. Diese Zustände können durch die Betrachtung der zugeordneten Ressourcen und der aus der Planung

generierten Ablaufzeiten identifiziert werden. Entsprechende parallele Abläufe werden anschließend in sequentielle Abläufe geändert. Mögliche Reihenfolgen werden als Alternativen eigetragen.

5.4.9 Definition der Anforderungen an Wissensquellen der Spezialistenebene

Die detaillierte Ausgestaltung aller möglichen Wissensquellen auf der Spezialistenebene ist nicht Zielsetzung dieser Arbeit. In diesem Abschnitt sollen jedoch die Anforderungen an die Wissensquellen beschrieben werden. HUMBURGER (1998) beschreibt darüber hinaus eine Methode zur Gestaltung von Wissensquellen, welche bei der Konzeption Anwendung finden kann. Zunächst müssen die Wissensquellen der Spezialistenebene mit der Strategie kommunizieren können. Dazu müssen sie zum einen ihre Ausführbarkeit eigenständig melden, damit sie aufgerufen werden können. Zum anderen müssen sie der Strategiewissensquelle Rückmeldung über ihren Erfolg oder Misserfolg oder gefundene Fehler geben. Für die Optimierung müssen die Wissensquellen der Spezialistenebene die notwendigen Werte an den Skill-Steps hinterlegen. Infolge der auftretenden Abhängigkeiten bei der Übergabe von Bauteilen müssen Wissensquellen, die Übergabeinformationen erhalten, diese interpretieren können. Sollten sich im Planungsprozess alternative Möglichkeiten durch unterschiedliche Produktpositionen und Orientierungen ergeben, so müssen diese alle in den SFC eingefügt werden. Insgesamt sind also prozesszentrierte Wissensquellen zu implementieren, die jedoch auch weiter hierarchisch aufgebaut sein können, um Funktionalitäten wiederzuverwenden. Eine Vorgabe ist, dass die für Primärprozesse notwendigen parallelen Funktionen, wie z. B. das Einspannen von Bauteilen in einem synchronisierten Ablauf zusammen mit dem Primärprozess, ausgeplant werden. Wird der SFC mit den Skill-Steps generiert, sind auch die zugehörigen Ressourcen abzulegen.

5.4.10 Beschreibung des Planungsprozesses an einem Beispiel

Zum besseren Verständnis werden die wesentlichen Schritte im Planungsprozess an einem Beispiel erläutert. Die Ausplanung der Prozesse durch die Wissensquellen der Spezialistenebene sowie die Optimierung anhand eines Kriteriums sind nicht vollumfänglich beschrieben. Abbildung 40 zeigt die zugehörige beispielhafte Montagezelle.

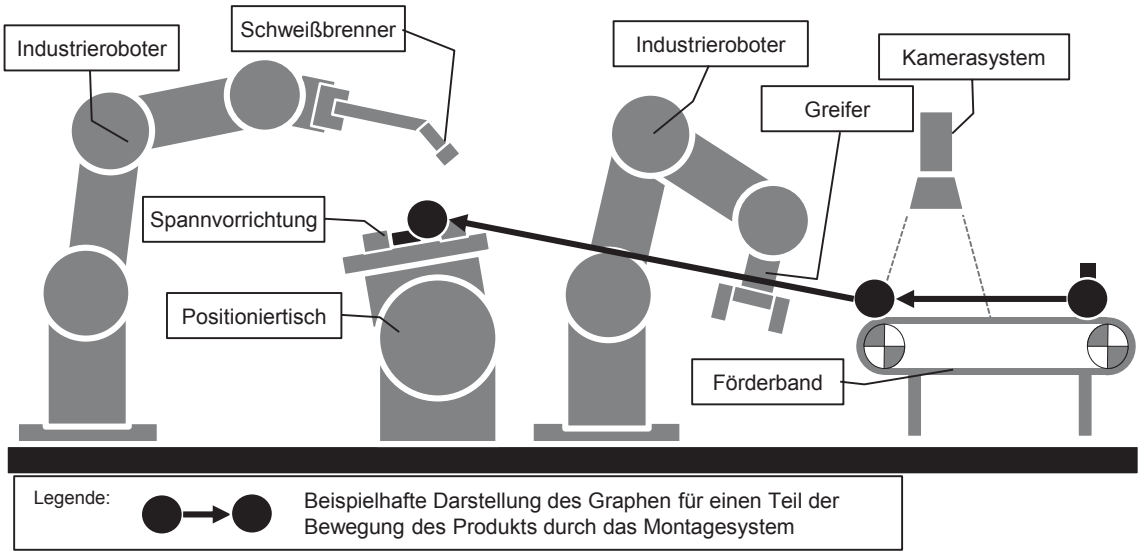


Abbildung 40: Beispielhafte Montagezelle zu Erläuterung des Planungsprozesses

In der Montagezelle werden Bauteile auf einen DKT verschweißt und dafür zunächst von einem Förderband aufgenommen und in der Spannvorrichtung abgelegt werden sollen. Für den Handhabungs- und Schweißvorgang steht jeweils ein Industrieroboter zur Verfügung. Ein Kamerasystem ermöglicht die Detektion der Bauteile auf dem Förderband.

Im ersten Schritt der Planung (vgl. Abbildung 41) erfolgt die Zuordnung der Ressourcen zu den Prozessen. Dies erfolgt über den Abgleich der in den Prozessen definierten Skills. Im betrachteten Beispiel sind das die Skills für die Bewegung, das Schweißen und das Einspannen.

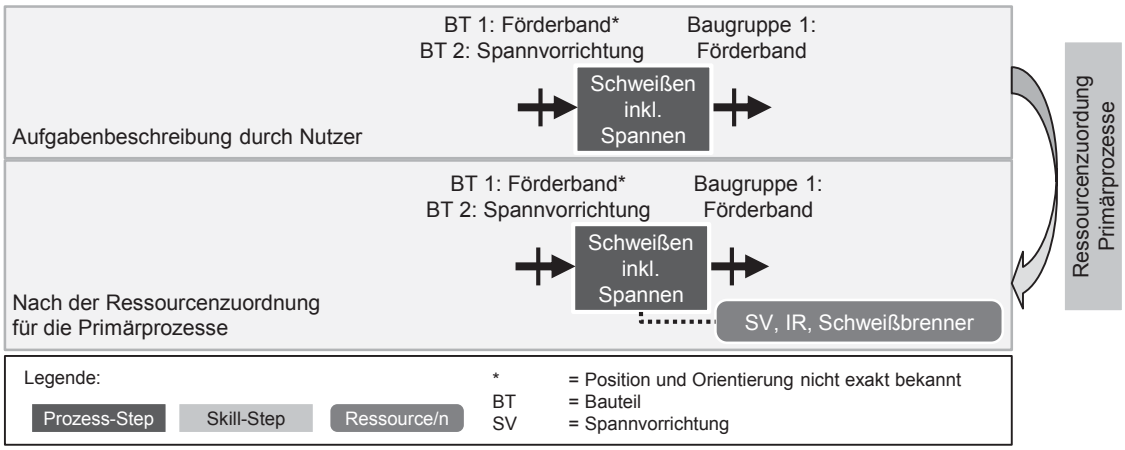


Abbildung 41: Darstellung des ersten Schritts der Planung an einem Beispiel

Durch die Zuordnung der Ressourcen kann im nächsten Schritt (vgl. Abbildung 42) durch die zugehörige Wissensquelle gefolgert werden, welche Sekundärpro-

zesse notwendig sind. Dabei werden auch Zwischenpositionen, wie z. B. der Arbeitsbereich der Spannvorrichtung zugeordnet. Nach der Identifikation aller notwendigen Sekundärprozesse erfolgt auch für diese die Zuordnung der Ressourcen unter Verwendung der in den Prozessen definierten Skills. Ist dieser Planungszustand erreicht, kann die Ausplanung durch die Wissensquellen der Spezialistenebene erfolgen.

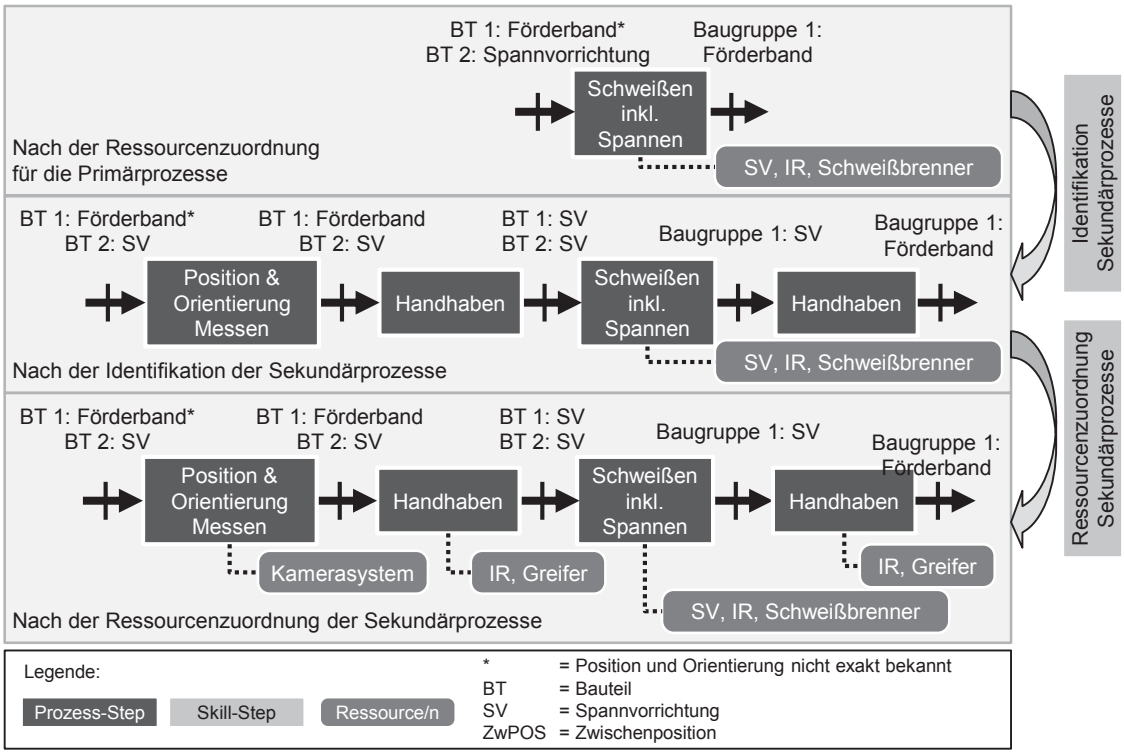


Abbildung 42: Darstellung des zweiten und dritten Schritts der Planung an einem Beispiel

Da die Ausplanung für alle vorkommenden Prozesse den Umfang der Darstellung übersteigen würde, ist nur ein ausgewählter Teil erläutert (vgl. Abbildung 43). In dem dargestellten Beispiel wurde der Primärprozess Schweißen ausgeplant. Anschließend wird der Sekundärprozess Handhaben geplant. Für die Handhabung der Bauteile ergeben sich Alternativen, z. B. die Zwischenpositionen in Abhängigkeit des Greifpunktes am Bauteil. Diese Alternativen werden in den SFC eingetragen. Einzelne Skill-Steps haben den Prozess-Step ersetzt. Auffällig ist, dass noch eine Abhängigkeit in der Sequenz der Skill-Steps vorhanden ist. Das sichern der Bauteile sollte ersten beendet werden, wenn diese eingespannt sind. Durch die zugehörige Wissensquelle zur Identifikation der Abhängigkeiten, kann dieser Zustand erkannt und der SFC umgeordnet werden. Der zugehörige Schritt ist nicht dargestellt.

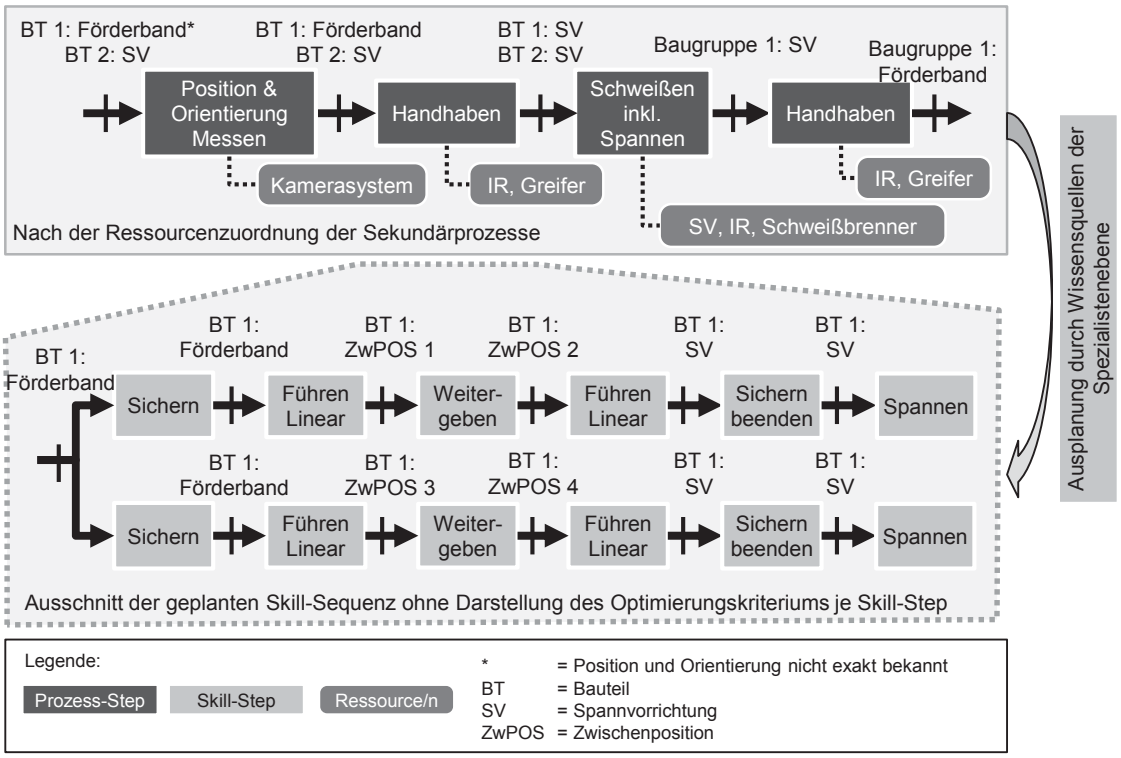


Abbildung 43: Darstellung eines ausgewählten Schritts der Planung an einem Beispiel

Wurden alle Prozesse erfolgreich ausgeplant, alle Abhängigkeiten behoben und etwaige Zusatzbewegungen eingefügt, kann die abschließende Optimierung des Ablaufs erfolgen (vgl. Abbildung 44). Im betrachteten Beispiel wird eine der möglichen Alternativen für die Handhabung ausgewählt.

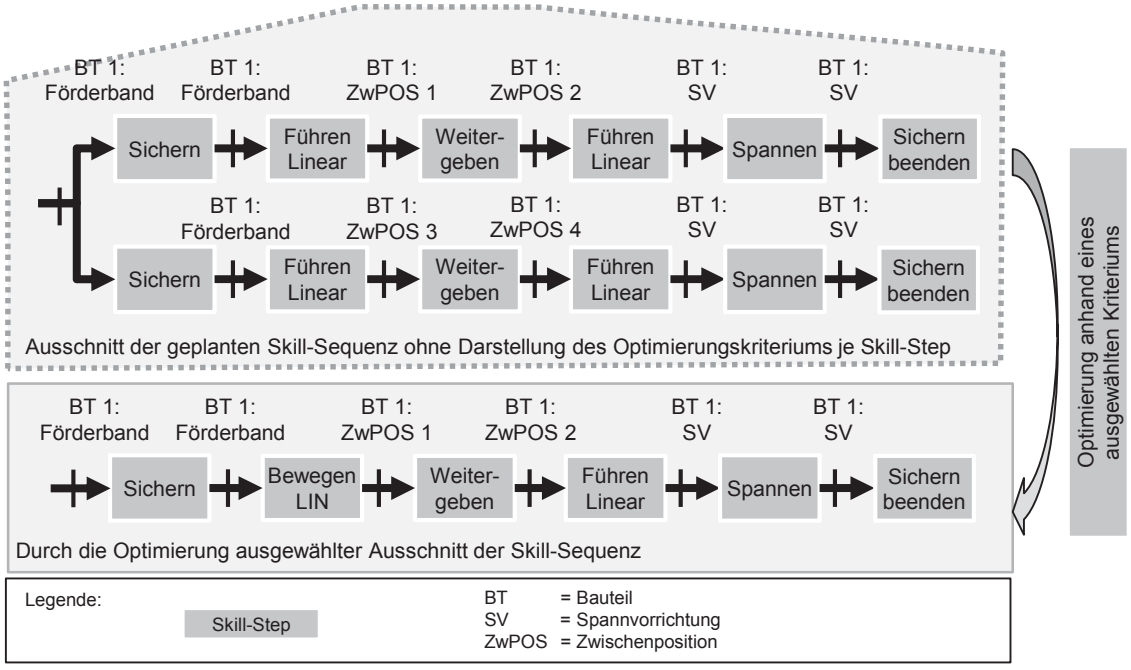


Abbildung 44: Darstellung eines Teils der Optimierung während der Planung

5.5 Konzeption weiterer Bestandteile des aufgabenorientierten Programmiersystems

In dieser Arbeit werden die weiteren Bestandteile des aufgabenorientierten Programmiersystems so weit spezifiziert, wie diese übergreifend anwendbar sind. Dieses Unterkapitel behandelt die Strukturierung von Nutzerschnittstellen, den Postprozessor sowie die Einbindung von Simulationssystemen.

5.5.1 Strukturierung der Benutzerschnittstellen für die aufgabenorientierte Programmierung von Montagesystemen

Die Relevanz der Eingabesysteme bzw. Benutzerschnittstellen für die einfache Programmierung, unterstreicht eine Vielzahl an Arbeiten in diesem Bereich (ZÄH ET AL. 2004; EHRMANN 2007; VOGL 2009; LAMBRECHT & KRÜGER 2014). Auf Grundlagen der Gestaltung von Nutzerschnittstellen wird an dieser Stelle nicht eingegangen. Es sei auf die entsprechende Literatur verwiesen (RICHTER & FLÜCKIGER 2007; SHNEIDERMAN & PLAISANT 2009). Grundsätzlich wird davon ausgegangen, dass sowohl einfache *Nutzer* als auch *Experten* mit dem Programmiersystem interagieren. Die Aufgabe der Experten ist es, Fehler zu beseitigen oder die Funktionalitäten des Systems für neue Prozesse oder Zielsysteme zu erweitern. Er arbeitet nicht produktiv mit dem System und wird daher im Folgenden nicht bei der Unterscheidung betrachtet. Die anschließenden Beschreibungen beziehen sich auf die einfachen Nutzer bzw. Endanwender. Nachstehend wird eine Strukturierung der Benutzerschnittstellen vorgestellt, welche die Auswahl bzw. die Implementierung einer Technologie und eines Konzepts für einen spezifischen Fall erleichtert. Für die aufgabenorientierte Programmierung müssen im Besonderen die Teilbereiche *Nutzergruppen*, einzugebende *Informationen bzw. Informationsarten* und *technische Konzepte für Nutzerschnittstellen* zur Strukturierung berücksichtig werden. In diesem Spannungsfeld erfolgt die Auswahl bzw. Implementierung der Nutzerschnittstelle.

Die notwendigen *Informationsarten* eines Endanwenders lassen sich aus dem Informationsmodell bzw. aus den Ebenen des Aufgabenmodells ableiten. Für Experten des Programmiersystems ist zusätzlich noch die Anpassung der vordefinierten Prozesse relevant. Dies wird aber bei der Strukturierung nicht beachtet. Notwendige Eingaben des Nutzers können die folgenden Arten umfassen:

- Geometrie (z. B. zur Beschreibung von Fügeverbindungen)
- Ablauf (z. B. bei der Vorgabe einer Primärprozesssequenz)
- Parameter (z. B. Klebedruck, Raupendurchmesser einer Klebeverbindung)

- Bausteine zur Programmerstellung (z. B. vordefinierte Prozesse oder Skills)
- Verwaltung (z. B. zum Aufruf alter Aufgabenbeschreibungen)

Die *technischen Konzepte* lassen sich aus den grundsätzlichen Arten der Programmierung ableiten. Für eine aufgabenorientierte Offline-Programmierung von Montagesystemen sind daher textuelle, visuelle und graphische bzw. simulationsbasierte Verfahren zu unterscheiden (EHRMANN 2007, S. 26). Im Rahmen dieser Arbeit werden multi-modale Eingabesysteme, d. h. mit einer Interaktion über mehrere Kommunikationswege, wie sie z. B. von LAMBRECHT & KRÜGER (2014) vorgestellt werden, als weitere Kategorie betrachtet. Auch das Programming-by-Demonstration kann diesem Bereich zugeordnet werden. Im Gegensatz zu klassischen graphischen bzw. simulationsbasierten Verfahren erfolgt die Nutzereingabe hier nicht an einem Rechnerarbeitsplatz.

Da das Programmiersystem nicht für jedes einzelne Individuum angepasst werden kann, wird zur bestmöglichen Beschreibung der Nutzer die Einteilung in *Nutzergruppen* zur Hilfe genommen (EHRMANN 2007). Die Nutzer einer Gruppe zeichnen sich durch ähnliche oder identische Aufgaben und Eigenschaften aus (EHRMANN 2007, S. 31). Einteilungen von Nutzergruppen finden sich u. a. bei EHRMANN (2007) und DAMMERTZ (1996). Eine direkte Übernahme dieser Nutzergruppen ist jedoch nicht zielführend, da für diese Arbeit Servicetechniker keine direkten Anwender des Programmiersystems darstellen. Es werden in Anlehnung an EHRMANN (2007) die Nutzergruppen *Bediener, Anwender* und *Prozessexperte* unterschieden. Alle drei Gruppen werden nach den Kriterien Prozesswissen, Funktionswissen, Programmierkenntnisse und Erfahrung mit Simulationssystemen unterschieden (vgl. Tabelle 3).

Die Zuordnung im Bereich Programmierkenntnisse des Prozessexperten wurde jedoch geändert. Das letzte Kriterium wurde in Ergänzung zu den von EHRMANN (2007) Verwendeten definiert. Der Prozessexperte kennt sich mit den einzelnen Montageprozessen und Simulationssystemen aus, hat aber ansonsten nur weniger Vorkenntnisse. Der Anwender kennt sich zwar mit den Ressourcen in Montagesystemen aus, hat jedoch nur grundlegendes Wissens bzgl. der Programmierung. Diese Art von Nutzer ist häufig in KMU zu finden. Der Bediener arbeitet im täglichen Betrieb mit den Montagesystemen und ist maßgeblich für die Überwachung der Prozesse zuständig und hat daher wenig Erfahrung in der Programmierung (EHRMANN 2007, S. 34). Im Umgang mit Simulationssystemen ist er nicht geschult.

Tabelle 3: Nutzergruppen mit zugeordneten Kenntnisbereichen in Anlehnung an EHRMANN (2007)

Kenntniss-bereich / Nutzergruppe	Funktions-wissen	Prozess-wissen	Erfahrung Simulations-systeme	Programmier-kenntnisse
Bediener	◔	◑	○	◔
Anwender	◑	◑	◑	◑
Prozessexperte	◔	●	◑	◔

○ kaum vorhanden ◔ gering ◑ durchschnittlich ◕ umfangreich ● vollständig vorhanden

In Abhängigkeit von den Benutzergruppen, welche das aufgabenorientierte Programmiersystem bedienen sollen und mit Hilfe der notwendigen Informationsarten kann die Auswahl der technischen Konzepte erfolgen, welche die Grundlage für die Benutzerschnittstelle darstellen. Als Hilfestellung wurden geeignete Zuordnungen erstellt, welche in Anhang A4 abgebildet sind. Diese Betrachtung ist auch deswegen nötig, da sich nicht alle Eingabekonzepte für alle Informationsarten eignen. Beispielsweise sind Multi-Modale Lösungen für die Eingabe vieler Parameter nicht geeignet. In Abhängigkeit vom Ergebnis der Betrachtung können auch Konzepte zum Einsatz kommen, welche mehrere technische Ansätze kombinieren. Die Umsetzungsplattform wird nicht betrachtet und kann frei gewählt werden.

Über die Benutzeroberfläche erfolgt typischerweise auch die Koordination des gesamten Programmiersystems. Da diese Koordination in enger Abhängigkeit von der ausgewählten Technologie der Umsetzung angepasst werden muss, erfolgt nur eine kurze Beschreibung der wesentlichen Funktionalitäten. Für die Koordination sind u. a. die folgenden Aufgaben notwendig:

- Koordination des Gesamtablaufs zu Programmgenerierung zwischen Planungsmodul und Postprozessor.
- Verarbeitung von Ereignissen und Programmfehlern sowie Rückmeldung an die Benutzerschnittstelle.
- Bereitstellen von Funktionalitäten zur Nutzer- und Funktionsverwaltung durch Administratoren bzw. Experten.
- Funktionen zum Import der durch den erweiterten Konfigurationsmanager erzeugten Zustandsmodelle.
- Funktionen zum Aufruf alter Aufgabenbeschreibungen und erzeugter Programme.

5.5.2 Generierung und Test der Steuerungsprogramme

5.5.2.1 Konzeption der Struktur eines Postprozessors für die Generierung steuerungsspezifischer Programme

Die grundlegende Aufgabe des Postprozessors ist es, aus den in SFCs beschriebenen Ablaufsequenzen der Skills, lauffähige Programme für die Steuerungen des Montagesystems zu generieren. Postprozessoren werden in einer Reihe von Ansätzen beschrieben (LÜDEMANN-RAVIT 2005; EHRMANN 2007; FRIEDRICH 2010) und stellen daher nicht den Fokus dieser Arbeit dar. Es werden nur spezifische Eigenschaften beschrieben, die sich aus dem Anwendungsfall dieser Arbeit und dem Informationsmodell ergeben. Folgende Anforderungen muss der Postprozessor erfüllen:

- Der Postprozessor muss sowohl für Robotersteuerungen, als auch für SPS ausgelegt sein. Insbesondere muss auch die Generierung von Programmen für mehrere Steuerungen innerhalb einer Anlage möglich sein.
- Der Postprozessor muss eine allgemeingültige Struktur besitzen, die sich einfach an die Spezifika einer Sprache anpassen lässt (LÜDEMANN-RAVIT 2005, S. 51).
- Ergibt sich die Notwendigkeit von Kommunikations- und Synchronisationsabläufen zwischen verschiedenen Steuerungen, so sind diese aus der Ablaufsequenz zu extrahieren und zu generieren.

Das Konzept des Postprozessors nach EHRMANN (2007) bietet sich im Besonderen für diese Arbeit an, da dieses sowohl RC als SPS fokussiert und darüber hinaus für die aufgabenorientierte Programmierung konzipiert wurde. Es wurde daher als Schema für das Konzept in dieser Arbeit verwendet, auch wenn dieses keine Skills verwendet. Der angepasste Ablauf des Postprozessors ist in Abbildung 45 dargestellt. Der erste Schritt des Postprozessors ist das Einlesen der Skill-Ablaufsequenz inklusive der Skill-Steps, der Übergänge und der Verzweigungen inklusive der zugehörigen Eingangsgrößen bzw. Übergangsbedingungen. Mit den zugeordneten Ressourcen und dem Umweltmodell kann zu jedem Skill die zugehörige *Steuerung zugeordnet* werden, worauf im nächsten Schritt das *Aufteilen der Skillsequenz* auf die einzelnen Steuerungen möglich ist. Anschließend können über die Pointer der instanziierten Skills die zugehörigen *Funktionen, Befehle oder Schnittstellen ausgelesen* werden. Dazu werden über die Schnittstellenobjekte die referenzierten Elemente im AutomationML-File identifiziert. Im nächsten Schritt werden die Codemodule für die Initialisierung sowie

Lese- und Schreibzugriffe von *Schnittstellen* wie Ethernet *erstellt* (EHRMANN 2007, S. 106). Dies umfasst auch die Generierung der Elemente zu Kommunikation und Synchronisation im Ablauf. Das eigentliche Programm wird im nächsten Schritt durch die *Transformation der Skillsequenz* in einzelne Codemodule generiert. Im Gegensatz zu einer SPS ist es bei einer Robotersteuerung nötig, *Positionsvariablen* zu *generieren* (EHRMANN 2007, S. 106), die aus den Eingangsgrößen der Skills für Bewegungen abgeleitet werden können. Für die *Erstellung der Hauptprogramme* werden die einzelnen Codemodule der Ressourcen mit Zustandsvariablen zu einem Ablauf zusammengefügt und Elemente zur Initialisierung aus der Ressourcenbeschreibung ergänzt. Globale Parameter mit den zugehörigen Datentypen werden im letzten Schritt zu einer *Parameterliste zusammengefügt* (EHRMANN 2007, S. 106).

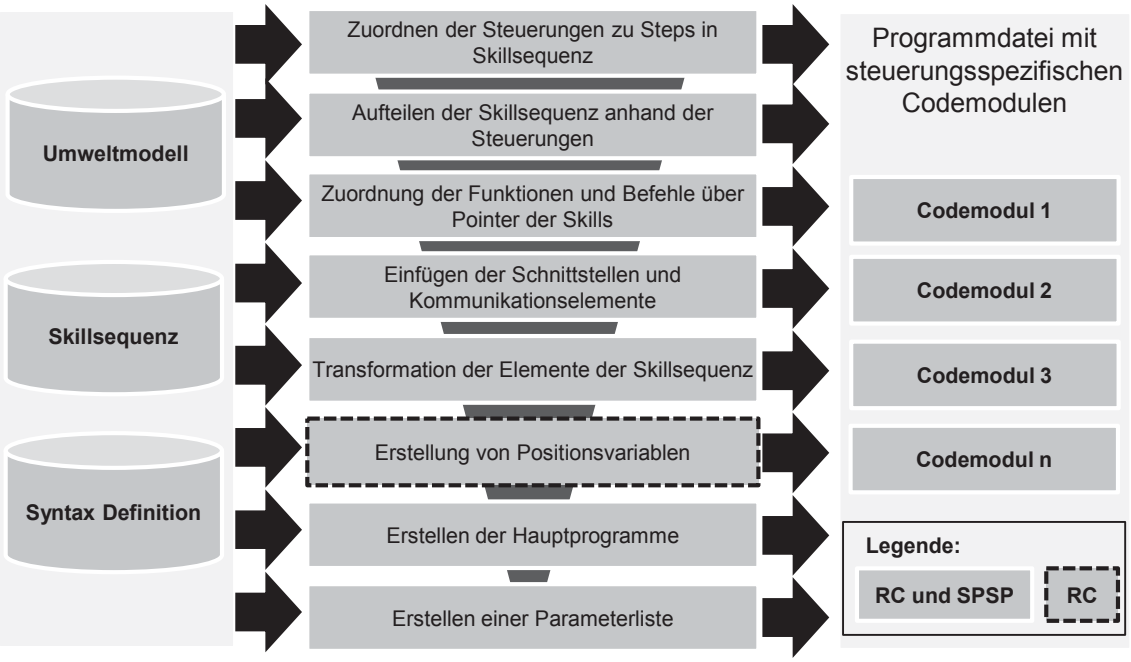


Abbildung 45: Ablauf und Bestandteile des Postprozessors (Eigene Darstellung in Anlehnung an EHRMANN (2007))

Für die Generierung der Programmdatei sind neben dem Umweltmodell, mit dem darin abgebildeten Mapping der Skills zu Befehlen und Schnittstellen sowie der Skill-Ablaufsequenz, auch die Syntax-Definitionen der Zielsysteme notwendig. Diese müssen für alle verwendeten Steuerungen vorhanden sein und beinhalten typischerweise Variablen mit Datentypen und Gültigkeitsbereichen, Deklarationen und Felder unbeschränkter Größe, Operatoren z. B. zur Berechnung und Routinen wie bedingte Anweisungen, Sprunganweisungen und Schleifen (LÜDEMANN-RAVIT 2005). Die Programmdatei setzt sich, wie im Ablauf beschrie-

ben, aus den Codemodulen für Schnittstellen und der Ablaufsequenz, der Positionsvariablen und der Parameterliste zusammen. Diese sind in ein Hauptprogramm eingebunden, welches über Zustandsvariablen den Ablauf festlegt (EHRMANN 2007, S. 106).

5.5.2.2 Test der erzeugten Steuerungsprogramme

Für eine realistische Abbildung des Steuerungsverhaltens, insbesondere bei Robotern, ist die Nachbildung der Steuerung in der Simulation notwendig (LÜDEMANN-RAVIT 2005, S. 28). Für Roboter ist dies z. B. mit der Realistic Robot Simulation (RRS) möglich (BERNHARDT ET AL. 1995). Eine Simulation für SPS-Code ist z. B. in CODESYS integriert (3S-SMART 2014). In jedem Fall sollte abschließend eine Simulation des erzeugten Steuerungsprogramms in einem Simulationssystem inklusive virtueller Steuerung stattfinden, um mögliche Programmfehler aufdecken zu können.

5.5.3 Integration von Simulationssystemen

Simulationssysteme, die im Programmiersystem zum Einsatz kommen, können bzgl. ihrer Abbildung der Steuerungen unterschieden werden. Für die schnelle Simulation vieler Möglichkeiten kann es im Planungsprozess sinnvoll sein, ein Softwaresystem ohne virtuelle Steuerung zu verwenden, um die Berechnungszeit des Planungsprozesses zu verkürzen. Für den Test der generierten Steuerungsprogramme ist jedoch die Abbildung der Steuerung in der Simulation notwendig. Einige kommerzielle Tools wie Process Simulate oder herstellerspezifische Programme wie KUKA.Sim bieten diese Möglichkeit. Sollen kommerzielle Systeme angebunden werden, deren Schnittstellen typischerweise eingeschränkt sind (HAMMERSTINGL & REINHART 2015), so bietet es sich an, in der Ressourcenbeschreibung auch die herstellerspezifischen Simulationsmodelle zu referenzieren und abzulegen. Die Simulationsanwendung kann entweder direkt in das Programmiersystem integriert oder über Programmierschnittstellen, sogenannte APIs (Application Programming Interface), angebunden werden. Die Schnittstelle beinhaltet typischerweise folgende Funktionalitäten: Generierung der Simulationsmodelle, Starten, Steuern und Überwachen von Simulation sowie Rückmeldung und Aufbereitung der Ergebnisse. Da die notwendigen Simulationssysteme und deren Anbindung stark vom Anwendungsfall des Programmiersystems abhängen, wird deren Ausgestaltung nicht weiter beschrieben.

5.6 Zusammenfassung

Nach einem Überblick über das entwickelte adaptierbare, aufgabenorientierte Programmiersystem wurde in diesem Kapitel zunächst das zugrundeliegende Informationsmodell beschrieben. Das Informationsmodell setzt sich aus den Teilbereichen Aufgaben- und Umweltmodell zusammen. Das Aufgabenmodell beinhaltet die Prozesse mit den notwendigen Skills, die zugehörigen Produkte sowie die Prozesssequenz. Es kann in sechs hierarchische Ebenen eingeteilt werden. Das Umweltmodell besteht aus den Bestandteilen Ressource und Skill. Der Fokus in dieser Arbeit lag auf der Herleitung der unterschiedlichen Klassen an Skills mithilfe eines Vorgehens. Ein weiterer Schwerpunkt wurde auf die Modellierung der Skills sowie deren Abbildung auf die Befehle und Kommunikationsschnittstellen von Ressourcen gelegt.

Im zweiten Teil dieses Kapitels wurde der Kern des entwickelten Programmiersystems, das Planungsmodul, beschreiben. Dieses basiert auf einer Blackboard-Architektur und ermöglicht damit eine hohe Flexibilität bzgl. Änderungen oder Erweiterungen. Indem das Planungsmodul auf dem Informationsmodell bzw. dem Konzept der Skills aufbaut, wird die Flexibilität weiter erhöht. So können verschiedene Komplexitätsebenen bei der Aufgabenbeschreibung durch den Nutzer verarbeitet werden, ohne das Programmiersystem ändern zu müssen. Neben der Struktur des Planungsmoduls wurden darüber hinaus Funktionalitäten beschrieben, welche unabhängig von den betrachteten Montageprozessen verwendet werden können. Abschließend wurden weitere Elemente des Programmiersystems, wie z. B. die Nutzerschnittstelle, beschrieben bzw. strukturiert.

Zusammen mit dem Informationsmodell wurde eine adaptierbare Grundlage geschaffen, um das aufgabenorientierte Programmiersystem mit geringem Aufwand für ein konkretes Montagesystem zu verwenden. Das Informationsmodell sowie die Beschreibung des Programmiersystems bilden die Grundlage für eine Umsetzung und Erprobung.

6 Vorgehensmodell zur Integration eines aufgabenorientierten Programmiersystems in den Engineeringprozess eines Unternehmens

6.1 Auswahl einer Grundlage für das Vorgehensmodell

Mit einem Vorgehensmodell ist es möglich, Unternehmen methodisch zu unterstützen, die an der Einführung der aufgabenorientierten Programmierung in ihre Engineeringprozesse interessiert sind. Damit werden diese in die Lage versetzt, die Einsatzpotentiale schnell abzuschätzen und das Programmiersystem so anzupassen, dass nur geringe Aufwände entstehen, die dieses Potential schmälern.

Wie in Abschnitt 3.6 beschrieben, existieren in der Literatur eine Reihe von Modellen und Vorgehen für die allgemeine Einführung von Softwaresystemen in Unternehmen. In diesem Abschnitt soll basierend auf den Anforderungen aus Kapitel 4 eine Grundlage für das Vorgehensmodell ausgewählt werden. Mit dieser können allgemeine Inhalte bzgl. dem organisatorischem Vorgehen und der Projektgestaltung abgedeckt werden, um darauf aufbauend die Inhalte für den Anwendungsfall der aufgabenorientierten Programmierung neu zu erarbeiten. Damit kann bzgl. der rahmengebenden Inhalte des Vorgehensmodells auf erprobte Ansätze aufgebaut werden. Die in Unterkapitel 3.6 vorgestellten Ansätze wurden daher bzgl. ihrer Eignung für die Integration bzw. Einführung eines aufgabenorientierten Programmiersystems bewertet (vgl. Tabelle 4).

Tabelle 4: Bewertung bestehender Ansätze zur Einführung von Softwaresystemen

Bewertungskriterien \ Ansätze in der Literatur (○ nicht erfüllt, ◑ teilweise erfüllt, ● erfüllt)	Stahn und Wilmerstaedt 1991	VDI-Richtlinie 2219	VDI-Richtlinie 4499	Koch 2005	Ghoffrani 2007	Nedbal 2013
Rahmenwerk für Organisation und Projektmanagement bei der Einführung von Softwaresystemen	◑	●	●	●	◑	●
Fokus auf betriebliche Informations- bzw. Erzeuger und Autorensysteme	●	○	◑	◑	○	◑
Beschreibung von Aktivitäten	◑	●	●	●	●	●
Beschreibung von Methoden	◑	●	◑	●	◑	●
Beschreibung von Werkzeugen	○	◑	◑	●	◑	●

Die Bewertung ergibt, dass insbesondere die Vorgehensmodelle von KOCH (2005) und NEDBAL (2013) für die Verwendung in dieser Arbeit geeignet sind. Die inhaltlichen Aktivitäten der VDI-RICHTLINIE 2219 sind außerdem zum Großteil in dem Konzept von NEDBAL (2013) zu finden. Da zudem Rollen, Aktivitäten, Methoden und Werkzeuge klarer im Modell eingeordnet werden, wird dessen Vorgehensmodell als Grundlage verwendet. Inhaltlich erfolgt eine Anpassung des Vorgehensmodells, wobei auch relevante Inhalte aus anderen Ansätzen zur Einführung von Softwaresystemen integriert werden.

6.2 Übersicht des angepassten Vorgehensmodells

Abbildung 46 zeigt die Übersicht des angepassten Vorgehensmodells für die Integration der aufgabenorientierten Programmierung in den Engineeringprozess eines Unternehmens.

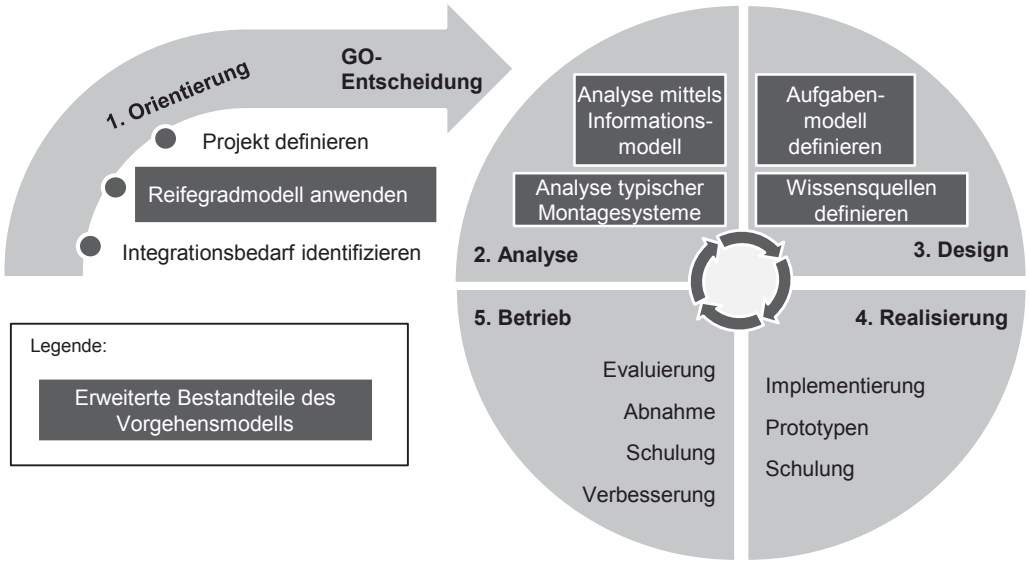


Abbildung 46: Vorgehensmodells für die Integration der aufgabenorientierten Programmierung in den Engineeringprozess eines Unternehmens in Anlehnung an NEDBAL (2013)

Das Modell wird in diesem Abschnitt zunächst im Überblick beschrieben, bevor in den folgenden Abschnitten die Kernelemente genauer erläutert werden. Grundsätzlich wird das Vorgehensmodell in zwei Stufen detailliert. Neben dem allgemeingültigen Modell für die generelle Integration der aufgabenorientierten Programmierung, wird das Modell für das in dieser Arbeit beschriebene adaptierbare Programmiersystem detailliert. In der ersten Phase, der *Orientierung*, sollen vorab die Potentiale der aufgabenorientierten Programmierung für das

Unternehmen eingeschätzt und anschließend das Projekt definiert werden. Für die Einschätzung der Potentiale wurde im Rahmen dieser Arbeit ein Reifegradmodell entwickelt. In der *Analyse* werden die Randbedingungen im Unternehmen detailliert aufgenommen. Der Fokus liegt auf den vorhandenen Daten sowie den bisherigen Geschäftsprozessen. Das Ergebnis der Analyse stellt die Anforderungsspezifikation dar (NEDBAL 2013). Im dritten Schritt, dem *Design*, werden aus den Anforderungen die notwendigen Inhalte des Programmiersystems festgelegt. Dies beinhaltet Nutzerrollen, Datenquellen und Zielsysteme sowie die funktionalen Inhalte des Planungsmoduls. Zusätzlich werden auch die geänderten Geschäftsprozesse definiert. In der *Realisierung* wird das Programmiersystem ausgewählt oder aufgebaut bzw. angepasst. Hierbei werden die notwendigen Funktionalitäten implementiert bzw. ergänzt. Darüber hinaus sind die neu definierten Geschäftsprozesse zu integrieren. Im *Betrieb* erfolgt die Abnahme, Evaluierung und kontinuierliche Verbesserung (NEDBAL 2013). Die von NEDBAL (2013) definierten Rollen bei der Einführung werden größtenteils übernommen. Dies sind der Projektleiter, der Koordinator, der Umsetzungspartner, das Management, das Kernteam, die Beta-Benutzer, die Endbenutzer und der Administrator bzw. Experte. Die betrachteten Integrationsebenen von NEDBAL (2013), *Daten*, *betriebliche Informationssysteme, Middleware, soziale Ebene* sowie die Ebene der *Prozesse und Services* werden ebenfalls übernommen. Im Rahmen dieser Arbeit werden unter der sozialen Ebene auch die Mitarbeiter mit deren Qualifikationen und Anforderungen aufgefasst.

6.3 Orientierungsphase bei der Einführung

Im Rahmen dieser Arbeit ist die Anwendung eines Reifegradmodells in der Orientierungsphase vorgesehen, um die aktuelle Situation zu bewerten und Potentiale aufzuzeigen. Das Reifegradmodell soll als Ordnungsrahmen dienen, um ein Unternehmen mit geringem Aufwand bzgl. der Eignung für die Einführung der aufgabenorientierten Programmierung einordnen zu können. Den Abschluss der Orientierung bilden die *Vorstellung der Potentiale* und die *Definition eines Projektes* (NEDBAL 2013). Zusammen mit den Potentialen sollte eine grobe Aufwand-Nutzen-Bewertung durchgeführt werden. Die erste Phase unterscheidet nicht zwischen dem allgemeinen Einsatz der aufgabenorientierten Programmierung und dem entwickelten adaptierbaren Programmiersystem.

6.3.1 Vorüberlegungen für ein Reifegradmodell für die aufgabenorientierte Programmierung

Bisher existiert kein Reifegradmodell für die aufgabenorientierte Programmierung. Es existieren jedoch Reifegradmodelle für unterschiedliche andere Anwendungsgebiete wie z. B. für mechatronische Entwicklungsprozesse (RAUCHENBERGER 2010), für Werkzeuglandschaften von IT-Service-Management-Prozessen (RICHTER 2013) oder für das Projektmanagement (AHLEMANN ET AL. 2005). Das Capability Maturity Model Integration (CMMI) beinhaltet darüber hinaus eine Sammlung von Reifegradmodellen für unterschiedliche Disziplinen (HOFFMANN 2013). Aus diesen können jedoch Grundkonzepte übernommen werden. Vor der Anwendung des Reifegradmodells in der Durchführung des Vorgehensmodells wurde ein angepasstes Reifegradmodell im Rahmen dieser Arbeit entwickelt.

Um das Reifegradmodell zu entwickelt, wurde ein Vorgehen angewendet, welches sich an dem Modell zur Entwicklung von Reifegradmodellen nach KNACKSTEDT ET AL. (2009) orientiert. Im ersten Schritt wurde daher das Problem definiert, bestehende Reifegradmodelle untersucht und daraus Anforderungen an die Modellstruktur und Inhalte abgeleitet. Wesentliche Anforderungen an das zu entwickelnde Reifgradmodell sind die Abbildung aller relevanten Teilaspekte eines Unternehmens durch eine Systematisierung des Betrachtungsraums sowie die aufwandsarme Anwendbarkeit. Im zweiten Schritt wurden die aktuellen Konzepte zur aufgabenorientierten Programmierung aus Kapitel 3 analysiert, um daraus im dritten Schritt die Inhalte und die Struktur des Modells ableiten zu können. Mit einer prototypischen Anwendung wurde das Modell im letzten Schritt weiter verbessert.

6.3.2 Übersicht des Reifegradmodells

Das in der Orientierungsphase anzuwendende Reifegradmodell, welches im Rahmen dieser Arbeit entwickelt wurde, lehnt sich in seiner Struktur an den Aufbau des Modells nach RAUCHENBERGER (2010) an und besteht aus Fragen und einer Berechnungsvorschrift, welche die Einordnung eines Unternehmens in die Stufen des Reifegradmodells ermöglichen. Das Modell von RAUCHENBERGER (2010) wurde für mechatronische Entwicklungsprozesse entwickelt und zielt somit im Grundsatz auf ein ähnliches Themengebiet. Der grundsätzliche Aufbau des Reifegradmodells wurde daher übernommen, dessen Inhalte wurden jedoch neu entwickelt. Wesentlicher Aspekt für den Einsatz einer aufgabenorientierten

Programmierung sind die notwendigen Daten für das Umwelt- und Aufgabenmodell und somit das *Datenmanagement*. Dieses wird von den *Geschäftsprozessen* und *sonstigen Randbedingungen* flankiert, wie z. B. der Mitarbeiterqualifikation. Zusammen stellen diese drei die Themenschwerpunkte des Reifegradmodells dar, die sich aus den verdichteten Integrationsebenen des Vorgehensmodells ableiten. Da ein aufgabenorientiertes Programmiersystem prinzipiell losgelöst von einem Engineering- bzw. Montageplanungsprozess eingesetzt werden kann, kommt dem Datenmanagement im Vergleich zum Entwicklungs- oder Geschäftsprozess eine besonders große Bedeutung zu. Allen drei Themenschwerpunkten sind Fragen zugeordnet, mit denen die jeweilige Situation bzw. das jeweilige Unternehmen bewertet werden kann. Die Fragen im Bereich des Datenmanagements leiten sich aus den Kernpunkten des in Abschnitt 5.3 beschriebenen Informationsmodells ab und sind jeweils Stufen des Reifegradmodells zugeordnet. Das detaillierte Reifegradmodell ist in Anhang A3 beschrieben.

Das Reifegradmodell umfasst insgesamt vier Stufen. Diese orientieren sich grob an den Evolutionsphasen der Produktentwicklung (SPUR 2001) sowie den in der VDI/VDE 3695 beschriebenen Zielzuständen des Anlagenengineerings bzgl. Beschreibungssprachen und dem Einsatz von IT-Werkzeugen. Die letzte Stufe entspricht dem durchgängigen Einsatz des Prinzips der digitalen Fabrik in einem Unternehmen.

1. Zeichnungsorientierte Entwicklung: In diesem Reifegrad liegen nur 2D-Zeichnungen der zu montierenden Bauteile vor, weitere Informationen bzgl. der Produkte sind nicht digital dokumentiert. Die Entwicklung des Produktes und des Montagesystems erfolgt typischerweise sequentiell und die Nutzer haben nur eine geringe Qualifikation im Umgang mit IT-Werkzeugen.

2. Rechnergestützte Entwicklung: Es liegen 3D-Geometriemodelle der Bauteile vor, welche typischerweise mit CAD-Systemen erstellt wurden. Weitere Informationen bzgl. des Produktes sind nicht digital dokumentiert. Es können Werkzeuge zur Konfiguration von Varianten zum Einsatz kommen. Die Planung des Montage- oder Fertigungsprozesses erfolgt nur für die Konstruktion digital, d. h. es liegen nur 3D-CAD Daten des Montagesystems vor. Weitere Informationen bzgl. des Montagesystems oder der angestrebten Prozessreihenfolge sind nicht maschinenlesbar und z. B. in Office Anwendungen abgebildet.

3. Rechnerorientierte Produktmodellierung: Das Produkt wird durchgängig digital entwickelt und simulativ abgesichert. Neben der 3D-Geometrie werden auch alle weiteren Eigenschaften des Produktes in einem digitalen Produktmodell mit

zugehöriger Beschreibungssprache verwaltet. Diese Modelle sind typischerweise in ein PDM- bzw. PLM-System integriert. Die notwendigen Montageprozessschritte sind nur teilweise digital beschrieben. Im Planungsprozess des Montagesystems existieren einzelne digitale Werkzeuge zur Durchführung von Simulationsstudien mit den zugehörigen Datenmodellen. Die Datenmodelle des Montagesystems sind aber nicht durchgehend und bilden nicht alle für die aufgabenorientierte Programmierung notwendigen Facetten wie Verhalten und Skills ab. Der Entwicklungsprozess von Produkt und Montagesystem ist parallelisiert.

4. Virtuelle Produkt- und Fabrikentstehung: Es existieren durchgängige digitale Modelle und Beschreibungssprachen sowohl für das Produkt und Montagesystem, als auch auf Seite der Montageprozesse. Virtuelle Werkzeuge und Modelle werden auch für den Produktanlauf und für die Planung des Montageprozesses eingesetzt. Alle verwendeten Systeme sind über ein durchgängiges Datenmodell mit zugehörigen Schnittstellen verknüpft.

Wegen der Anforderung des geringen Aufwands, wird im Gegensatz zu RAUCHENBERGER (2010), nur eine Ebene an Fragen verwendet, die in die drei genannten Themenschwerpunkte unterteilt ist. Alle Fragen sind darüber hinaus mit Gewichtungsfaktoren beaufschlagt und mit einer Berechnungsvorschrift versehen. Insgesamt beinhaltet das Reifegradmodell 31 Fragen. Sollten Fragen für ein Untersuchungsziel nicht zutreffen, z. B. weil nur das Engineering des Montagesystems und nicht die Gestaltung des zu produzierenden Produktes in den Kernkompetenzen des Unternehmens liegt, so können diese übersprungen werden und fließen nicht in die Berechnung mit ein. Für die Anwendung, d. h. die Identifikation des Reifegrads für die aufgabenorientierte Programmierung, sind die Fragen der Teilbereiche strukturiert zu beantworten und anschließend das Ergebnis zu berechnen. Für die weitere Einführung ist es notwendig, dass das Unternehmen mindestens auf Stufe zwei des Reifegrads eingeordnet wird. Anderenfalls sind die zusätzlichen Aufwände bei der Einführung der aufgabenorientierten Programmierung zu hoch, um deren Nutzen zu rechtfertigen. Auch auf Stufe zwei ist das Kosten/Nutzen-Verhältnis jedoch kritisch zu prüfen. Unabhängig von der Einstufung können die Fragen jedoch verwendet werden, um Handlungsanweisungen abzuleiten. Die Fragen in den Teilbereichen werden an dieser Stelle nicht beschrieben, können aber in Anhang A3 nachgelesen werden. Dort sind auch die Formeln zur Berechnung des Reifegrads aus den Ergebnissen der Fragen dargestellt.

6.4 Analysephase bei der Einführung

Um die aktuelle Situation in Unternehmen zu identifizieren, schlägt NEDBAL (2013) die drei Aktivitäten *Ist-Analyse, Workshops* und *Erfolgsfaktorenanalyse* vor, die auch für diese Arbeit übernommen werden. Er beschreibt weitere anwendbare Methoden und Werkzeuge. Insbesondere die Ist-Analyse hat für die aufgabenorientierte Programmierung besondere Relevanz und wird daher weiter detailliert. Für die Integration sind insbesondere die Integrationsebenen *Daten, Prozesse, betriebliche Informationssysteme* und *soziale Ebene* von besonderer Relevanz. Zur Ist-Aufnahme in diesen Bereichen wird ein detailliertes Vorgehen vorgeschlagen.

Ausgangspunkt der Ist-Analyse sind die Unternehmensprozesse, welche aufzunehmen sind. Für diese können anerkannte Beschreibungsformen wie die Business Prozess Modeling Language (BPML) (WOHED ET AL. 2006), Informationsflusslandkarten, SADTs oder eEPKs angewendet werden. Hilfreiche Fragen zur Prozessaufnahme finden sich u. a. in der VDI-RICHTLINIE 2219. In den Prozessen sind auch die verwendeten Informationssysteme inklusive der Nutzerrollen mit abzubilden, wie es z. B. mit eEPK möglich ist. Die Qualifikation der Mitarbeiter im späteren Einsatzbereich des aufgabenorientierten Programmiersystems ist im nächsten Schritt detailliert aufzunehmen. Einteilungsmöglichkeiten für Nutzer in die Gruppen Bediener, Anwender, oder Prozessexperte wurden in Abschnitt 5.5 vorgestellt. Im dritten Schritt kann das in dieser Arbeit entwickelte Informationsmodell mit allen Teilbereichen herangezogen werden, um die Daten detailliert zu betrachten. Dabei ist jeweils zu klären, ob die Daten vorhanden sind und wenn ja, in welchen Formaten bzw. Beschreibungssprachen. Übergreifend sind die datenverarbeitenden Systeme und deren Schnittstellen aufzunehmen. Für die allgemeine Überprüfung können die Skills im Informationsmodell übergangen werden. Diese sind nur für die Einführung des adaptierbaren Programmiersystems dieser Arbeit relevant. Im vierten und letzten Schritt werden bestehende Montagesysteme bzgl. der darin vorkommenden Montageprozesse und Betriebsmittel analysiert und deren Arten, Zuordnung und Häufigkeit dokumentiert. Die Einteilung der Montageprozesse erfolgt anhand der bekannten VDI- und DIN-Normen. Die Zusammenfassung dieser Analyseergebnisse bildet wie beschrieben die Anforderungsspezifikation.

6.5 Designphase bei der Einführung

Im Design wird spezifiziert, wie das aufgabenorientierte Programmiersystem und die neu gestalteten Geschäftsprozesse gestaltet werden sollen. Nach der VDI-RICHTLINIE 2219 sollte in dieser Phase die Umgestaltung und ideale Definition der Geschäftsprozesse der erste Schritt sein. KOCH (2005) gibt für die Gestaltung der Kernprozesse umfangreiche Hilfestellung. Beispielsweise kann dies auch bedeuten neue Schritte im Geschäftsprozess zu definieren, um die notwendigen Daten für das Programmiersystem zu erzeugen. Bei der Gestaltung der Geschäftsprozesse kann grundsätzlich eine Makro- (bezogen auf das gesamte Unternehmen) und eine Mirkoebene (bezogen auf Steuerungsprojektierung bzw. Softwareentwicklung) unterschieden werden, welche beide auszugestalten sind. Soll die Auswahl aus verschiedenen kommerziellen Tools erfolgen, so ist eine Bewertung durchzuführen. Ein Vorgehen zur Auswahl, inklusive wirtschaftlicher Bewertung, ist beispielsweise in der VDI-RICHTLINIE 2219 beschrieben. Das Ergebnis ist entweder ein Design-Dokument (NEDBAL 2013) oder das Pflichtenheft bei der Auswahl aus bestehenden kommerziellen Systemen (VDI-RICHTLINIE 2219). Die folgende Beschreibung beschränkt sich daher nur auf Aspekte, welche direkt für die aufgabenorientierte Programmierung relevant sind und noch nicht beschrieben wurden.

Mit der Festlegung der Nutzerrollen in den Geschäftsprozessen können direkt die Anforderungen an die Nutzerschnittstelle des Programmiersystems abgeleitet werden. Je nach Anforderung können z. B. klassische graphische Benutzeroberflächen oder Augmented-Reality-Konzepte zielführend sein. Zusammen mit den bestehenden Daten aus dem Entwicklungsprozess legen die Nutzerrollen das Aufgabenmodell des Programmiersystems fest. Für das adaptierbare Programmiersystem bedeutet dies konkret die Festlegung der Ebene im Aufgabenmodell. Mit dem in der Analyse identifizierten Spektrum an Montageprozessen kann das Aufgabenmodell spezifiziert werden, indem notwendige Prozesse, inklusive deren Eingabeparameter, durch den Experten definiert werden. Zielführend ist es, die Prozesse sowohl im Programmiersystem als auch in der Planung der Montageprozesse zu verwenden und die entsprechenden Geschäftsprozesse anzupassen. In einem ersten Entwurf des Systems eignet sich eine Auswahl der Montageprozesse, die dem Pareto-Prinzip folgt. Für das aufgabenorientierte Programmiersystem bedeutet dies die Definition der Prozesse mit den zugehörigen Skills. Dabei fließt auch die Analyse der bisherigen Montagesysteme mit ein. Aus den Datenquellen und -formaten im Geschäftsprozess sowie den Zielsystemen aus den

bisherigen Montagesystemen, können notwendige Treiber sowie Funktionalitäten des Postprozessors abgeleitet werden.

Mit den oben genannten Informationen können im letzten Schritt die eigentlichen Funktionalitäten des Programmiersystems bzw. des Planungsmoduls abgeleitet werden. Dabei wird definiert, für welche Montageprozesse Planungsfunktionalitäten notwendig sind und welche Teilbereiche, wie z. B. Bahnplanung für 6-Achs-Industrieroboter, diese enthalten. Je nach Anforderungen werden auch Simulationssysteme eingebunden. Bezogen auf das adaptierbare Programmiersystem bedeutet dies die Definition der Wissensquellen, wobei die Methode zur Wissensquellendefinition nach HUMBURGER (1998) Anwendung finden sollte.

6.6 Realisierungsphase bei der Einführung

In der Realisierung werden die angepassten Geschäftsprozesse eingeführt, verankert und das aufgabenorientierte Programmiersystem implementiert bzw. adaptiert. Dessen Ausgestaltung hängt vom System und den gewählten internen oder externen Umsetzungspartnern ab. NEDBAL (2013) gibt Hinweise, wie die Realisierung im Idealfall ablaufen sollte. Als Ergebnis dieser Phase sollte ein Prototyp des aufgabenorientierten Programmiersystems bestehen, mit dem erste Schulungen durchgeführt wurden. In Bezug auf das aufgabenorientierte Programmiersystem sind somit die Wissensquellen und die Benutzerschnittstelle zu implementieren sowie der Umfang des Postprozessors bei Bedarf zu erweitern.

6.7 Betriebsphase bei der Einführung

Nach der Produktivsetzung des aufgabenorientierten Programmiersystems sollten die wesentlichen auch von NEDBAL (2013) genannten Aktivitäten, wie Evaluation, Abnahme, Weiterentwicklung und weitere Nutzerschulungen erfolgen. Weiterentwicklungen in Bezug auf die aufgabenorientierte Programmierung sind insbesondere die Erweiterung der Planungsfunktionalitäten für weitere Montageprozesse und andere Zielsysteme.

6.8 Zusammenfassung

Um den Einsatz eines aufgabenorientierten Programmiersystems zu erleichtern, wurde in diesem Kapitel ein Vorgehensmodell für dessen Integration in den Engineeringprozess entwickelt. Das Vorgehensmodell baut auf einem bestehen-

den Ansatz auf und somit fokussiert sich die Beschreibung auf die speziellen Randbedingungen der aufgabenorientierten Programmierung. Allgemeine Aspekte bzgl. des Projektmanagements werden durch das Basismodell abgedeckt. Kernaspekt des Vorgehensmodells ist ein Reifegradmodell. Mit den darin enthaltenen Fragen können Unternehmen einfach abschätzten, inwieweit sich der Einsatz eines aufgabenorientierten Programmiersystems bei ihren jeweiligen Randbedingungen eignet.

7 Umsetzung und Erprobung

7.1 Aufbau des Kapitels

Dieses Kapitel teilt sich in zwei wesentliche Bestandteile. Im ersten Abschnitt wird die anwendungsfallunabhängige Softwareumsetzung des adaptierbaren Programmiersystems beschrieben. Dies beinhaltet jedoch noch nicht die Implementierung von Wissensquellen der Spezialistenebene für einzelne Anwendungsfälle, sondern nur die übergreifend anwendbaren Bestandteile. Im zweiten Abschnitt werden die betrachteten Referenzszenarien für die Erprobung beschrieben, das Vorgehensmodell exemplarisch auf eines davon angewandt und die implementierten Funktionalitäten im Programmiersystem dargelegt.

7.2 Softwaretechnischer Umsetzung eines Grundaufbau des aufgabenorientierten Programmiersystems

Der softwaretechnische Grundaufbau erfolgte unter vorab definierten Rahmenbedingungen. Da für alle Anwendungsfälle eine einheitliche Benutzeroberfläche verwendet wird, erfolgt deren Beschreibung auch in diesem Abschnitt. Für die Anwendungsbeispiele wird vorab festgelegt, dass die Beschreibung der Aufgabe durch den Nutzer auf Montagesequenz-, Sekundärprozess- oder Skillebene erfolgt und damit auch die Funktionalitäten zur Ableitung der Montagereihenfolge nicht notwendig sind. Die softwaretechnische Umsetzung folgt der 3-Schichten-Architektur aus Abbildung 17.

Präsentationsschicht - Benutzeroberfläche:

Die Benutzeroberfläche wurde als Webserver für Google Chrome, unter Verwendung des MEAN-Konzepts (MongoDB, Express, Angular.js, Node.js) (HAVIV 2014), umgesetzt. Dies erlaubt die Verwendung der Benutzeroberfläche auf verschiedenen Clients und ist systemunabhängig. Die Oberfläche hat für Experten und Anwender getrennte Bereiche, so dass durch den Experten Erweiterungen z. B. für neue Prozesse vorgenommen werden können. In der Oberfläche kann der Anwender einen Prozessablauf oder eine Skill-Sequenz mit vordefinierten Bausteinen per Drag & Drop beschreiben sowie den aktuellen Aufbau des Montagesystems in einer 3D-Visualisierung betrachten. Über Schaltflächen lassen sich der SFC der Aufgabenbeschreibung erzeugen sowie die eigentliche

Planung starten. Abbildung 47 zeigt den Aufbau und die Bestandteile der implementierten Benutzeroberfläche.

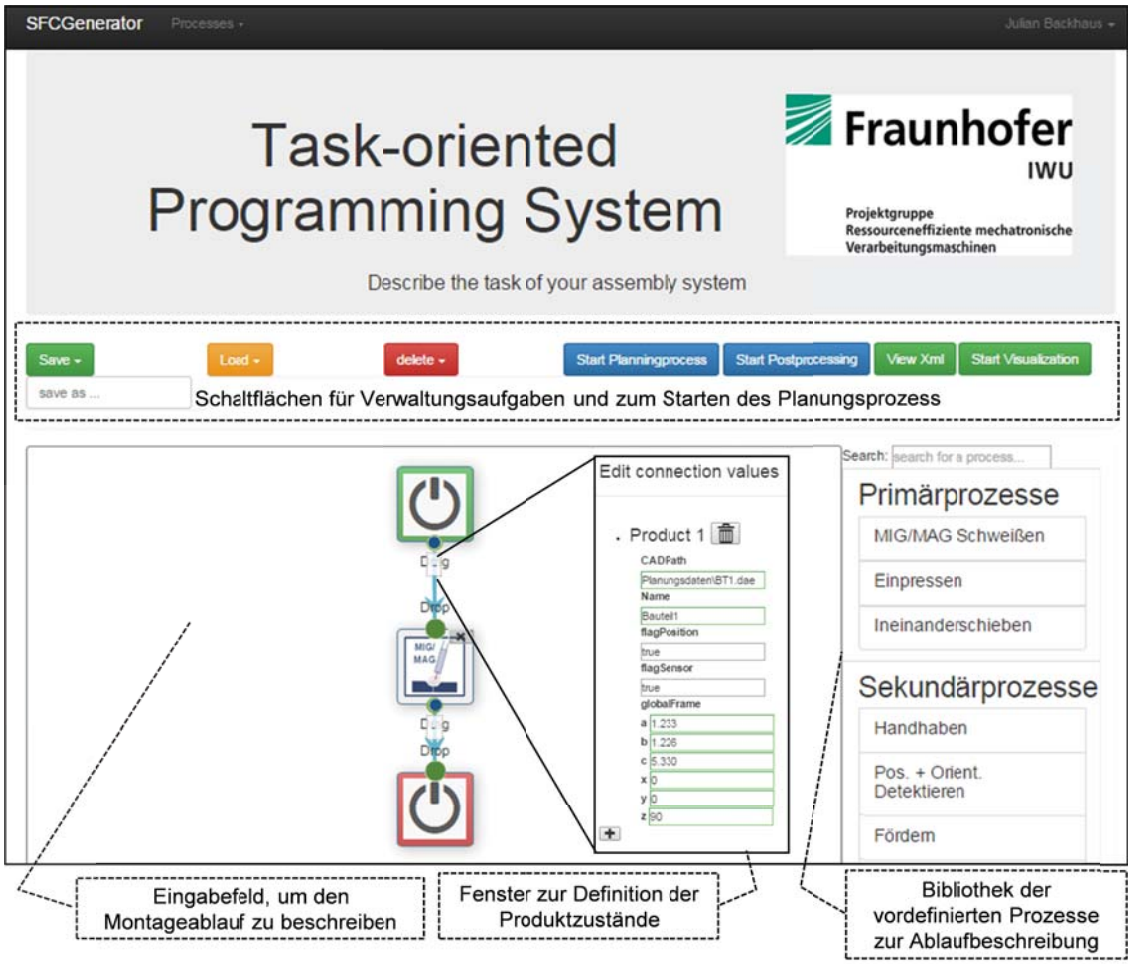


Abbildung 47: Webbasierte Benutzerobefläche des aufgabenorientierten Programmiersystems

An den Transitionen zwischen den Prozessschritten können die Produktbeschreibungen hinterlegt und die Parameter der Prozesse können ebenfalls in einem weiteren Fenster festgelegt werden. Sowohl in der Benutzeroberfläche, als auch im Planungsmodul werden einheitliche SFC XML Bibliotheken verwendet, welche sowohl Templates der Prozess- und Skill-Schritte, als auch für Transitionen enthalten. Änderungen können damit durch Austausch der Bibliotheken übergreifend durchgeführt werden.

Logikschicht

Wesentlicher Kern der Logikschicht ist das Planungsmodul. Dessen Grundaufbau inklusive der zugehörigen übergreifenden Wissensquellen, welche in Abschnitt 5.3 beschrieben sind, wurde in C++ implementiert. Die Klassenstruktur der

Softwareumsetzung folgt der Struktur des Planungsmoduls (vgl. Abbildung 35) und ist daher an dieser Stelle nicht dargestellt. Die Wissensquellen der Aufgabenebene sind als dynamische Bibliotheken (Dynamic Link Libraries, DLL) implementiert und können während der Ausführung dynamisch geladen werden. Vordefinierte virtuelle Funktionen ermöglichen einen standardisierten Zugriff auf diese. Dies ermöglicht eine einfache Erweiterung um neue Funktionalitäten insbesondere auf Spezialistenebene. Als Optimierungskriterium steht im Prototyp nur die Montage- bzw. Durchlaufzeit zur Verfügung. Diese zu minimieren ist ein wesentliches Ziel bei der Montageplanung und -gestaltung (LOTTER & WIENDAHL 2012). Die Grundstruktur des Postprozessors wurde auch in C++ umgesetzt. Nach erfolgreicher Planung kann dieser über die Nutzeroberfläche aufgerufen werden.

Datenschicht

Die Planung erfolgt durchgängig in AutomationML bzw. den zugehörigen Beschreibungsstandards wie PLCopen SFC XML. Systemdaten, wie die Kontrolldaten des Planungsmoduls, werden in internen Datenstrukturen gespeichert.

7.3 Anwendung in Referenzszenarios

Entsprechend der Zielsetzung wurden die entwickelten Konzepte dieser Arbeit sowohl für den Anwendungsfall der Erzeugung von Steuerungsprogrammen für eine Robotersteuerung, als auch für eine SPS angewendet. Das entwickelte Vorgehensmodell wurde nur im ersten Referenzszenario angewendet, da dieses unabhängig von der verwendeten Steuerungstechnologie ist.

7.3.1 Referenzszenario „Roboterbasiertes MIG/MAG Schweißen“

Beschreibung des Referenzszenarios

Als Versuchsumgebung des Referenzszenarios wird der von KRUG (2013) aufgebaute Versuchsstand verwendet (vgl. Abbildung 48). Dieser besteht aus einem Sechsachs Industrieroboter KUKA KR 15/2 mit einer KR C2 Steuerung, einem Drehkipptisch DKP-400 zur Positionierung sowie dem Funktionsprototyp einer Schweißsteuerung mit zugehörigem Schweißbrenner. Innerhalb der Zelle kommt als Echtzeit-Kommunikationsnetz Industrial-Ethernet Powerlink zum Einsatz (KRUG 2013). Die Schweißsteuerung und die Schweißbrenner können somit einen MIG/MAG (Metal-Inert-Gas/ Metal-Aktiv-Gas) Schweißprozess nachbil-

den. Die Schweißbrenner sind jedoch nicht voll funktionsfähig und zeigen nur über einen LED an, ob aktuell geschweißt wird.

Bei dem Unternehmen, in dem das aufgabenorientierte Programmiersystem fiktiv eingesetzt und das Vorgehensmodell angewendet werden soll, handelt es sich um einen Hersteller von elektrischen Antrieben mit ca. 500 Mitarbeitern. Digitale Daten der Produkte liegen umfassend vor und im Montageplanungsprozess werden digitale Beschreibungen des Montageablaufs und -systems erstellt. Für die beiden letztgenannten Beschreibungen werden vorrangig Office-Werkzeuge verwendet.

Abbildung 48: Versuchsstand des Referenzszenarios „Roboterbasiertes MIG/MAG Schweißen“ (KRUG 2013)

Exemplarische Anwendung des Vorgehensmodells zur Integration

Entsprechend dem Vorgehensmodell wurde im ersten Schritt der Orientierung das Reifegradmodell angewendet. Da das betrachtete Unternehmen die Reifegradstufe zwei erfüllt, ist der Einsatz der aufgabenorientierten Programmierung lohnenswert. Die Zuordnung erfolgte auf Grund der im vorherigen Abschnitt kurz skizzierten Ausprägung der Entwicklungsprozesse im Unternehmen. Im zweiten Schritt wurden die bestehenden Daten und Geschäftsprozesse genauer analysiert. Auf Basis der Analyse konnten die Anforderungen und die zu verwendende Prozessbeschreibung abgeleitet werden. Die verwendeten Skills wurden aus der in dieser Arbeit entwickelten Skill-Hierarchie den Prozessen zugeordnet. Dabei wurden die Skills „Führen Linear“, „Führen Circular“, „Weitergeben“ sowie der lösbare komplexe Skill „MIG/MAG Schweißen“ verwendet. Der

lösbare komplexe Skill ist weiter in „MIG/MAG Schweißen – Strom“, „MIG/MAG Schweißen – Gaszufuhr“, „MIG/MAG Schweißen – Drahtvorschub“ und „MIG/MAG Schweißen – Elektrode“ zerlegbar. Auf Grund der bestehenden Daten wurde eine Beschreibung der Aufgabe auf Primärprozessebene gewählt.

Wie im Informationsmodell definiert, besteht ein Prozess aus einem Geometrie- und Prozessanteil. Der Geometrieanteil beschreibt in diesem Fall die Form der Naht. Dazu wurde eine Beschreibung von Kehlnähten, V-Nähten, Halbe-V-Nähten, Y-Nähten, Halbe- Y-Nähten und I-Nähten erarbeitet und in COLLADA umgesetzt. Die Nahtgeometrie kann durch den Konstrukteur entweder über einen Postprozessor des CAD-Systems ausgeleitet oder durch die Verwendung von Templates der Nahttypen direkt definiert werden. Im vorliegenden Fall wurde, da kein Postprozessor vorhanden war, die zweite Möglichkeit verwendet. Der Prozessanteil des definierten Prozess-Steps besteht aus den Parametern Schweißgeschwindigkeit, Schweißstrom, Schweißspannung, Drahtvorschub, Schutzgasvolumenstrom und Art des Schutzgases, wobei nur die Art des Schutzgases zwingend vom Anwender vorgegeben werden muss. Für alle anderen Parameter wurde eine Datenbank verwendet, um diese automatisch aus den Angaben der Naht und des Bauteils abzuleiten. In der Realisierung wurde die Grundstruktur des aufgabenorientierten Programmiersystems erweitert. Die Beschreibung der implementierten Funktionalitäten erfolgt im nachfolgenden Abschnitt und wird daher an dieser Stelle nicht dargestellt. Die gewählte Ebene der Aufgabenbeschreibung macht nur eine geringe Anpassung der Geschäftsprozesse notwendig. Zum einen hat der Konstrukteur zusätzliche Aufgaben und muss die Prozess-Schritte mit zugehöriger Schweißnahtgeometrie definieren, zum anderen ist im Normallfall kein ausgebildeter Roboterprogrammierer notwendig, da die Erstellung des Roboterprogramms keine Experten mehr erfordert. Der Roboterprogrammierer kann also in anderen Gebieten des Unternehmens eingesetzt werden. Auf Grund des zwar real existierenden Unternehmens, aber fiktiven Anwendungsfalls, erfolgte keine Betrachtung der Betriebsphase des Programmiersystems.

Softwaretechnische Erweiterung des aufgabenorientierten Programmiersystems

Ausgehend von der Prozessbeschreibung wurden die notwendigen Funktionalitäten der Wissensquelle zur Planung eines roboterbasierten MIG/MAG

Schweißprozesses entworfen und implementiert. Die als .DDL implementierte Wissensquelle umfasst die folgenden Funktionalitäten:

- Ein- und Auslesen der Daten aus dem AML-File des Blackboards
- Planen der Schweißparameter, ausgehend von einer Schweißnahtbeschreibung
- Planen von erreichbaren Aufspannungen auf dem DKT
- Planen der notwendigen Schweißbewegung und der Transferbewegung, basierend auf der Schweißnahtgeometrie, einer Kollisions- und Erreichbarkeitsanalyse unter Verwendung des Simulationssystems V-REP (FREESE ET AL. 2010)

Eine Übersicht der Funktionalitäten und Schnittstellen zeigt Abbildung 49.

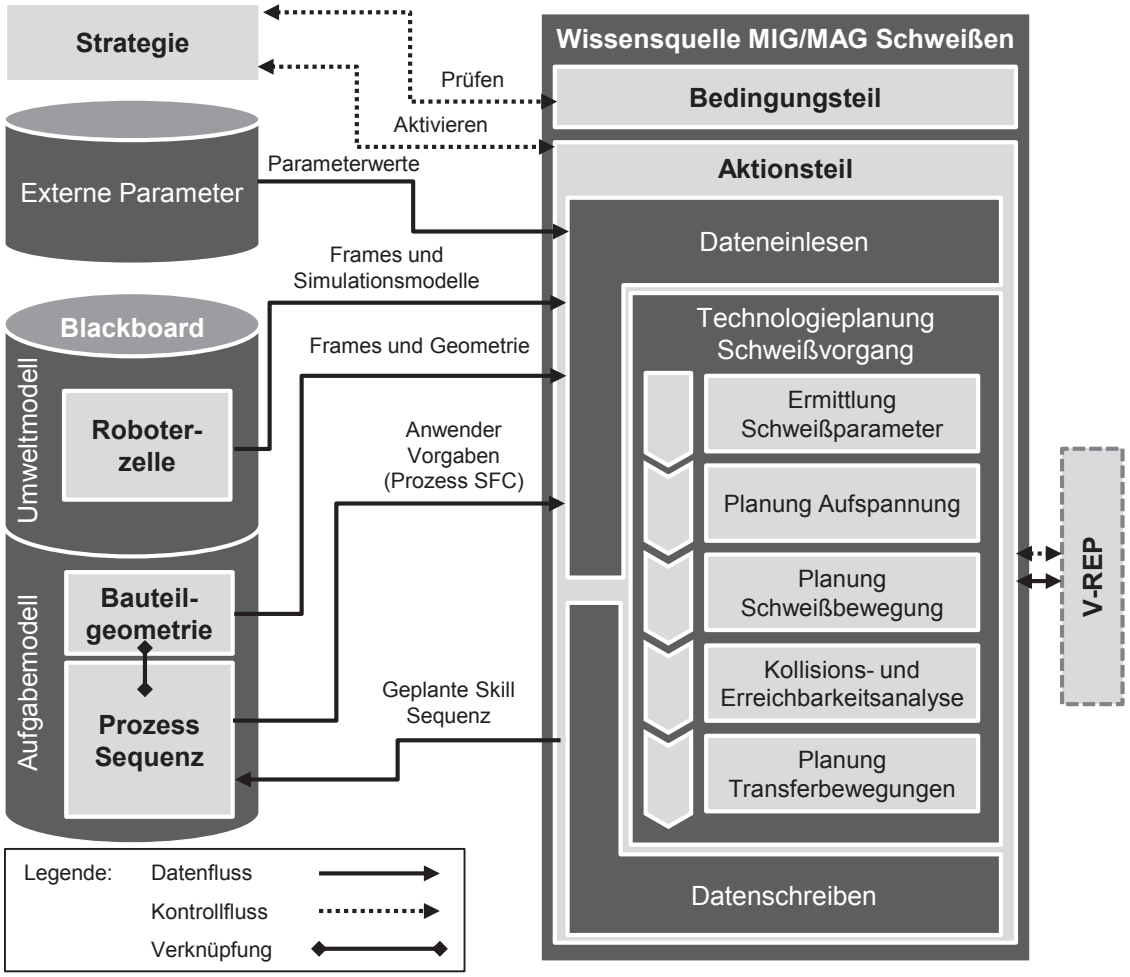


Abbildung 49: Funktionalitäten der Wissensquelle MIG/MAG Schweißen

Für die Generierung des Umweltmodells wurde der Konfigurationsmanager von KRUG (2013) erweitert und die Skillbeschreibung in die Gerätebeschreibungen

integriert. Zusätzlich wurde basierend auf dem Grundkonzept des Postprozessors dessen Funktionalität für die Generierung von KUKA KRL (KUKA Robot Language) erweitert. Dies umfasst die Generierung der notwendigen .dat und .src Dateien.

Ergebnisse der Erprobung

Vor der Anwendung mit Referenzbauteilen wurden die Layoutdaten der Roboterzelle und deren Bestandteile aufgenommen und das automatisch erzeugte Zustandsmodell damit erweitert. Für die Erprobung der Funktionalität wurden mehrere typische Referenzbauteile konstruiert und das Programmiersystem mit diesen getestet. Nach der Generierung der Steuerungsprogramme konnten diese erfolgreich am realen System getestet werden. Abbildung 50 zeigt die generierten Bahnen für eines der verwendeten Referenzbauteile.

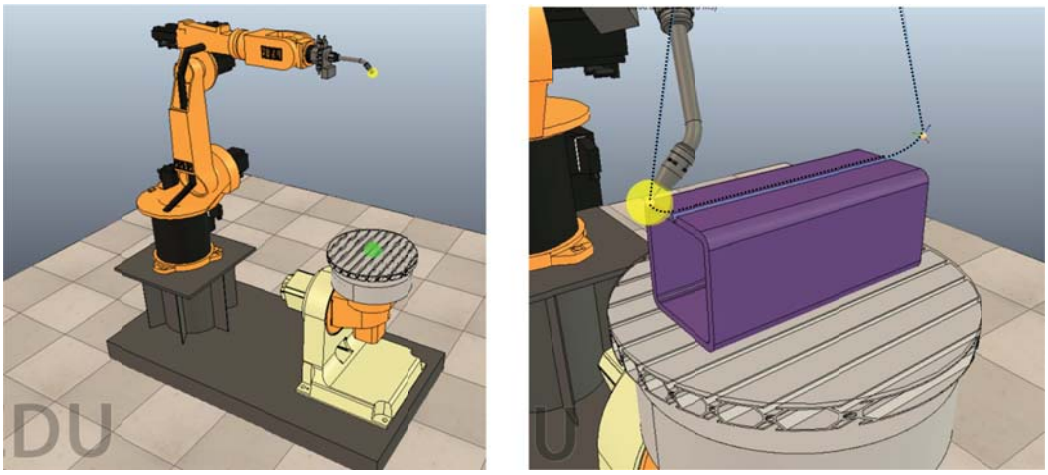


Abbildung 50: Simulationsmodell und generierte Bahnen des aufgabenorientierten Programmiersystems bei einer vorgegebenen V-Naht

Der Aufwand zur Programmierung wird durch das aufgabenorientierte Programmiersystem im aktuellen Aufbau gesenkt. Eine Versuchsgruppe, bestehend aus vier Personen ohne Spezialwissen in der Roboterprogrammierung, konnte die Programmierung für die Referenzbauteile nach kurzer Einweisung innerhalb von ca. zwei Minuten durchführen. Als Vergleich wurde das Teachen des Roboters in einer Offline-Simulation mit KUKA.Sim durchgeführt. Die Positionen und Orientierungen der Bauteile waren jeweils gegeben. Die Generierung des Umweltmodells erfolgte für das aufgabenorientierte Programmiersystem durch den erweiterten Konfigurationsmanager, wobei angenommen wurde, dass die Hersteller der Ressourcen die Modelle zur Verfügung stellen. Auch ohne Betrachtung der Zeit zur Modellierung der Roboterzelle benötigten die Probanden für die Pro-

grammierung mit KUKA.Sim ca. vier Minuten. Damit konnte exemplarisch gezeigt werden, dass eine Reduzierung der Zeit für die Programmierung durch das aufgabenorientierte Programmiersystem um ca. 50 % Prozent möglich ist. Da dies jedoch nur für den betrachteten Anwendungsfall gilt und die Einsparung in der Zeit für die Programmierung von der Ebene des Aufgabenmodells abhängt, wurde keine umfangreiche Untersuchung durchgeführt.

Simulative Erweiterung des Referenzszenarios

Um die Adaptierbarkeit des Konzepts auch für weitere Anwendungsfälle aufzeigen zu können, wurden weitere Wissensquellen auf Spezialistenebene konzipiert und implementiert, deren Ergebnisse jedoch nur simulativ überprüft wurden. Des Weiteren konnte damit auch die Funktionalität des Planungsmoduls bei abwechselnder Ausplanung von Primär- und Sekundärprozessen erprobt werden.

Zur Erweiterung des ersten Referenzszenarios wurden in einer Simulation zwei Förderbänder mit darüber montiertem Kamerasystem (Vision Sensor der Firma Festo) sowie ein weiterer Industrieroboter (ABB IRB 2600) mit pneumatischen Zweibackengreifer (Schunk PGN Plus 125) in der Zelle platziert (vgl. Abbildung 51).

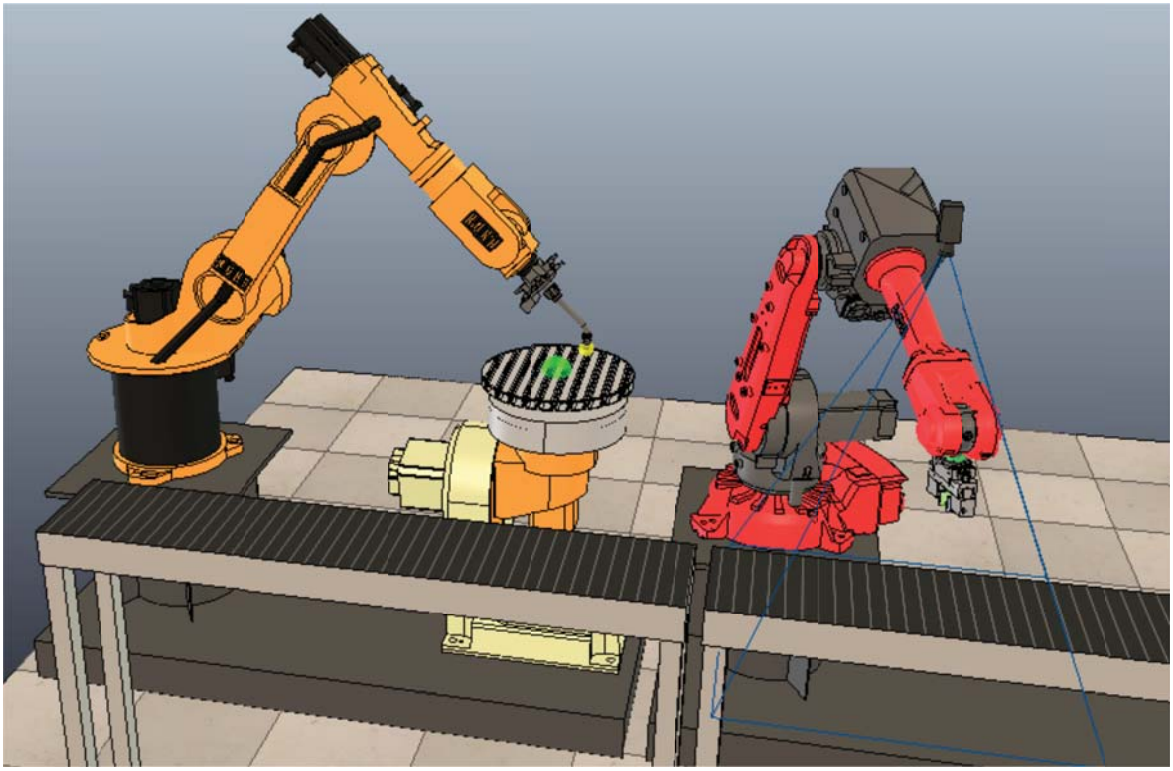


Abbildung 51: Simulationsmodell des erweiterten Referenzszenarios

Die zu schweißenden Bauteile sollen nun in ihrer Lage auf dem ersten Förderband erkannt, auf den DKT umgesetzt, dort verschweißt und anschließend auf das zweite Förderband zum Abtransport umgesetzt werden. Um für dieses Szenario Steuerungscode erzeugen zu können, sind somit Funktionalitäten für die Planung von Handhabungsbewegungen, als auch zur Einbindung der Förderbänder und des Kamerasystems notwendig. Für alle Bestandteile des Szenarios erfolgt die Modellierung entsprechend der im Informationsmodell beschriebenen Bestandteile in AutomationML.

Zur Planung des Handhabungsvorgangs wurde eine Wissensquelle bestehend aus einem Konzept zur Greifpunktbestimmung, aufbauend auf der Arbeit von RÖHRDANZ (1998) sowie einer Bahnplanung unter der Verwendung der Funktionalitäten von V-REP implementiert. Dabei werden die COLLADA Daten der Bauteile und des Greifers eingelesen und über mehrere Filter und geometrische Operationen Greifpunkte bestimmt, für welche anschließend die Bewegungsbahnen des Roboters abgeleitet werden können. Weitere Wissensquellen überprüfen die Einsetzbarkeit des Kamerasystems mit der Kontrolle von Hindernissen auf Basis der Sensorfeldgeometrie und dessen Strahlengang und planen den Einsatz der Förderbänder, um die Positionen von Bauteilen zu ändern. Mit dem generierten Skill-SFC und dessen manueller Überprüfung konnte gezeigt werden, dass die Funktionalitäten des Programmiersystems für den Anwendungsfall valide sind und insbesondere auch der Materialfluss in komplexeren Montagesystemen automatisiert geplant werden kann. Es konnte außerdem gezeigt werden, dass die gewählte Architektur zum einen einfach erweiterbar ist und zum anderen, dass das gewählte Informationsmodell eine passende Grundlage dafür bietet.

7.3.2 Referenzszenario „SPS-gesteuertes Montagemodul“

Beschreibung des Referenzszenarios

Für das zweite Referenzszenario wird als Versuchsumfeld ein Modul einer Modellfabrik am *iwb* verwendet, in welcher die Anlagentechnik inklusive Aktoren und Sensoren aus Fischertechnik abgebildet wird. Die Steuerung erfolgt über eine industriell eingesetzte Steuerung, in diesem Fall eine Beckhoff CX9020. Das Modul besteht aus einem Förderband, jeweils einer Lichtschranke an den Enden des Förderbandes sowie einer Vorrichtung zum Stempeln. Die Vorrichtung zum Stempeln besteht aus einem Motor sowie zwei Endschaltern. Abbildung 52 zeigt den Aufbau des Moduls.

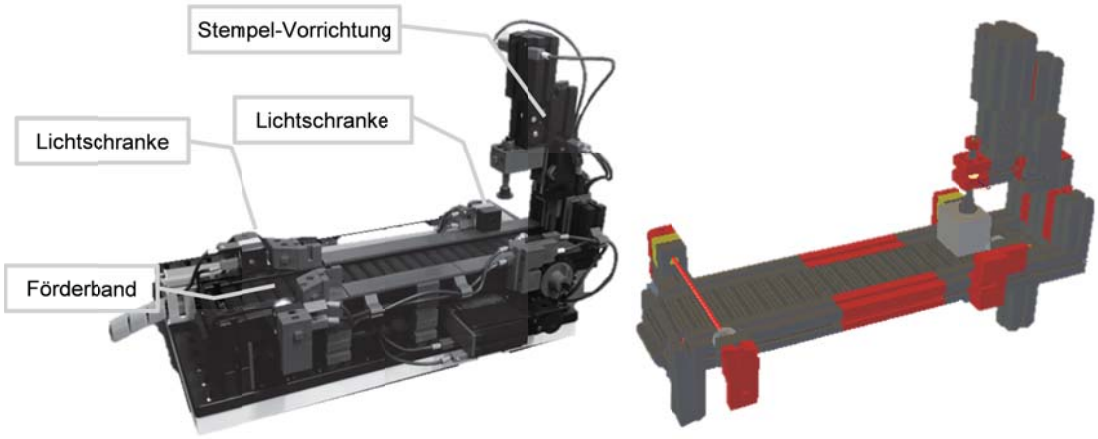


Abbildung 52: Versuchsanordnung und CAD-Modell des Referenzszenarios „SPS-gesteuertes Montagemodul“

Vorüberlegungen zur technischen Integration und Modellierung

Die Beschreibung der Aufgabe wird auf der Ebene der Sekundärprozesse festgelegt, da sich die notwendige Reihenfolge fest aus der Anlagenarchitektur ergibt. Entsprechend werden die Prozesse „Stempeln“ und „Fördern“ definiert. Dem Prozess-Step „Stempeln“ ist nur der Skill „Stempeln“ zugeordnet. Der Prozess-Step „Fördern“ hat die zugehörigen Skills „Fördern“ sowie „Anwesenheit Messen“, deren Zusammenwirken bei der Ausplanung des Prozesses durch eine Wissensquelle auf Spezialistenebene festgelegt werden soll. Da der Stempel nicht über Kommunikationsschnittstellen konfiguriert werden kann, muss der Nutzer keine zusätzlichen Parameter für den zugehörigen Prozess definieren.

Es wäre darüber hinaus nicht zielführend, einen spezifischen Skill für den beschriebenen Aufbau des Stempels zu verwenden. Dieser wäre nur für Stempelvorrichtungen gültig, die den gleichen Aufbau mit Endschaltern besitzen. Es wird daher ein abstrakter Skill für das Stempeln verwendet, welcher jedoch auch durch das Modul unterstützt werden muss. Die Umsetzung auf die Aktoren und Sensoren der Vorrichtung wird durch einen Funktionsablauf fest definiert. Diese Beschreibung könnte z. B. durch den Hersteller zur Verfügung gestellt werden und ist in der Ressourcenbeschreibung der Stempelvorrichtung in der Funktionsbeschreibung als PLCopen XML SFC hinterlegt. Das Mapping des Skills erfolgt also auf die Ein- und Ausgänge dieser Funktion.

Alle Ressourcen werden entsprechend den Vorgaben und Bestandteilen im Informationsmodell modelliert. Im Folgenden werden daher nur ausgesuchte Aspekte beschreiben. Bei der Modellierung der Lichtschranken ist insbesondere die

Modellierung der Arbeitsbereiche über die Angabe der Geometrie des Strahlengangs sowie die Abbildung des instanziierten Skills „Anwesenheit messen“ auf deren digitalen Ausgang relevant. Die Abbildung erfolgt, wie im Abschnitt 5.3 beschrieben, durch Pointer in AutomationML. Die Modellierung der Schnittstelle für das Förderband erfolgt nach dem gleichen Vorgehen. Bei dessen Abbildung im Umweltmodell ist insbesondere die Kopplung des Verhaltens- und Geometriemodells mit der Verknüpfung des Skills „Fördern“ bedeutsam. Nur über die Kopplung kann automatische auf die Bewegungsrichtung des Förderbands geschlossen werden.

Softwaretechnische Erweiterung des aufgabenorientierten Programmiersystems

Für die softwaretechnische Erweiterung wurden zusätzlich zur Grundstruktur des Planungsmoduls zwei Wissensquellen auf Spezialistenebene entworfen und der Postprozessor angepasst. Die Softwarestruktur des Programmiersystems ändert sich folglich nicht. Die beiden Wissensquellen erweitern die Blackboard-Architektur auf Spezialistenebene. Wegen der Kapselung der Stempel-Vorrichtung durch einen Funktionsablauf, muss die zugehörige Wissensquelle diesen nur durch den entsprechenden Skill-Step im SFC ersetzten. Die Wissensquelle für den Prozess „Fördern“ plant, unter Verwendung der Arbeitsbereiche des Förderbands und der Lichtschranken, den Transport der zu bedruckenden Gegenstände in den Arbeitsraum des Stempels und zurück in die Ausgangsposition. Die notwendige Logik, um das Zusammenspiel der Lichtschranken und des Förderbands zu koordinieren, ist also in der Wissensquelle implementiert. Die Geschwindigkeit des Förderbandes und die Abmessung der Produkte werden von der Wissensquelle außerdem zur Berechnung einer exakten Position, unabhängig von der Detektion der Lichtschranken, verwendet.

Für das SPS-gesteuerte Modul wurde der erweiterte Konfigurationsmanager nicht angepasst. Für das Anwendungsbeispiel war es daher notwendig, das zugehörige Projekt und die Hardwarekonfiguration manuell in TwinCAT 3 (BECKHOFF AUTOMATION 2015) anzulegen. Ein erweiterter Konfigurationsmanager könnte diese, basierend auf den Ressourcenbeschreibungen (insbesondere z. B. GSD-Dateien), automatisch erzeugen. Aufgabe des adaptierbaren Programmiersystems ist es, das Hauptprogramm (MAIN) und die notwendige Variablentabelle zu erzeugen. Diese müssen vom Nutzer anschließend in das SPS-Projekt importiert werden. Der Postprozessor wurde basierend auf dessen Grundaufbau für die Anforderungen der SPS angepasst. Da der Import von PLCopen XML SFC in

TwinCAT 3 direkt möglich ist, muss keine weitreichende Übersetzung des in der Planung generierten SFC erfolgen. Im SFC müssen jedoch die Skill-Steps über die Zuordnung durch die Pointer in Ein-, Ausgänge und Funktionen, entsprechend der Hardwarekonfiguration der Ressourcen, übersetzt werden. Die Konsistenz der Verdrahtung der Ressourcen mit der Hardwarekonfiguration sowie deren Repräsentanz im Umweltmodell wird daher vorausgesetzt.

Ergebnisse der Erprobung

Nach der Implementierung der notwendigen Softwarebestandteile in C++ erfolgte die Erpobung. Da die Zwischenschritte der Planung vor dem Nutzer verborgen und die Abbildung des zugehörigen SFC schwer verständlich ist, erfolgt eine graphische Darstellung der Schritte während der Planung. Durch die übergeordneten Wissensquellen der Aufgabenebene erfolgt die Ressourcenzuordnung und anschließend wird durch die Wissensquelle Stempeln der Prozess-Step Stempeln durch den zugehörigen Skill-Step ersetzt (vgl. Abbildung 53).

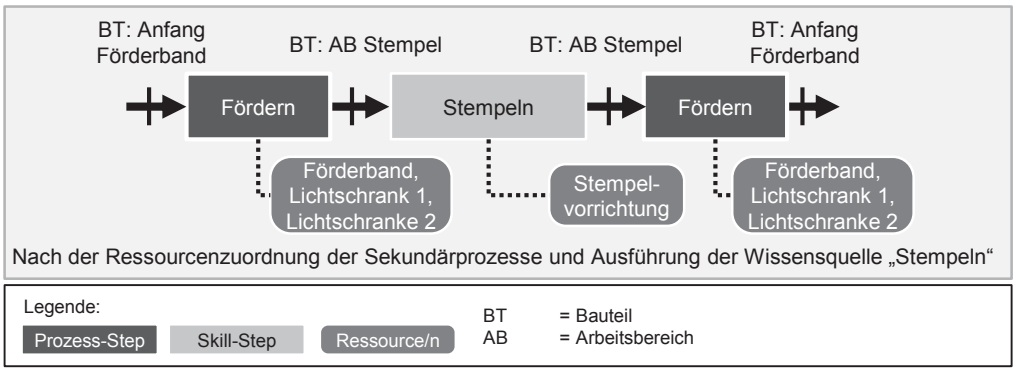


Abbildung 53: Zwischenstand der Planung für das SPS-gesteuerte Montagemodul

Bei der Ausplanung der Prozess-Steps zum Fördern muss nicht nur eine Abfolge an Skill-Steps generiert werden. Für das Starten bzw. Anhalten des Förderbands liefern die Lichtschranken die zugehörigen Kriterien. Da es sich beim Fördern um eine kontinuierliche Bewegung handelt, werden die beiden Skill-Steps „Fördern Start“ und „Fördern Stop“ verwendet. An der Transition vor und nach dem Skill-Step „Fördern Start“ wird zusätzlich eine zugehörige Condition definiert (vgl. Abbildung 54).

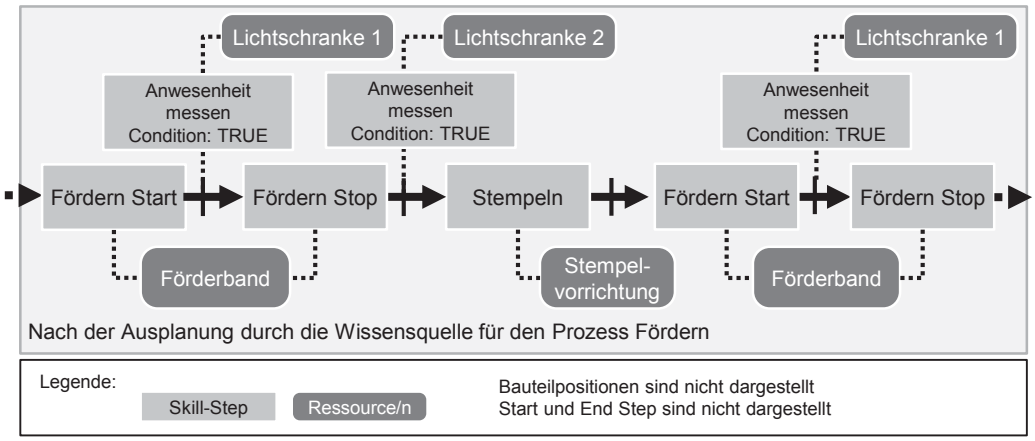


Abbildung 54: Zwischenstand nach Ausplanung aller Prozess-Steps

Die Spezifikation des SFC beinhaltet das zugehörige Element und die notwendigen Informationen werden über ein addData-Objekt hinterlegt (vgl. Abbildung 55). Die Zuordnung der Ressourcen erfolgt über die im Blackboard modellierten Arbeitsbereiche. Nach der Ausplanung des SFC kann im letzten Schritt die Übersetzung des SFCs in eine für die Steuerung interpretierbare Form erfolgen. Dabei wird der Skill-Step Stempeln durch den Postprozessor durch die zugehörige SFC-Sequenz in der Funktionsbeschreibung der Ressource ersetzt. Die Ausführung der Programme konnte, nach manueller Übertragung des Projektes auf die Steuerung, erfolgreich getestet werden. Somit konnte prototypisch gezeigt werden, dass das entwickelte aufgabenorientierte Programmiersystem mit dem zugehörigen Informationsmodell auch für SPS-gesteuerte Montagesysteme anwendbar ist.

```
<transition localId="2">
    ...
    <condition>
      <addData xmlns="">
        <data name="iwb">
          <iwb>
            <trigger nameRessource="Lichtschranke_1" nameSkill=„Anwesenheit_messen“ valueSkill="true"
          </iwb>
        </data>
      </addData>
    </condition>
    ...
</transition>
```

Abbildung 55: Modellierung der Condition mit Referenzierung der Skill-Steps in AutomationML

7.4 Zusammenfassung

Um die Erfüllung der gestellten Anforderungen und die Umsetzung der entwickelten Konzepte zu prüfen, wurde in diesem Kapitel ein Softwareprototyp des aufgabenorientierten Programmiersystems beschrieben. Nach dem Grundaufbau des Systems wurde es für zwei Anwendungsfälle adaptiert und mit den notwendigen Funktionalitäten ausgestattet. Zusätzlich wurden simulativ weitere Anwendungsfälle betrachtet und die notwendigen Funktionalitäten in Form von Wissensquellen implementiert. Dies umfasste insbesondere eine Wissensquelle mit Funktionalitäten zur Griffidentifikation und Bahnplanung für Handhabungsvorgänge.

Im ersten Anwendungsfall wurde das Programmiersystem für einen roboterbasierten Schweißprozess erweitert. Die notwendigen Funktionalitäten wurden in Form von Wissensquellen in die Blackboard-Architektur des Planungsmoduls eingegliedert. Mit einem angebundenen Simulationssystem konnten Funktionalitäten wie die Bahnplanung für einen Industrieroboter einfach umgesetzt werden. Der Postprozessor des Programmiersystems erzeugt einen ausführbaren Code für die verwendete Robotersteuerung. Ein beispielhafter Vergleich mit einem kommerziellen Programmiersystem konnte auch die erreichbaren Vorteile aufzeigen. Im zweiten Anwendungsfall wurde der Grundaufbau für ein SPS-gesteuertes Montagemodul angepasst. Auch hier wurden die notwendigen Funktionalitäten in Form von Wissensquellen integriert. Mit einem angepassten Postprozessor können ausführbare Programme erzeugt werden, welche in eine bestehende Konfiguration des Moduls in einer Projektierungssoftware integriert werden können. Mit der Umsetzung und Erprobung konnte die Plausibilität des verwendeten Informationsmodells sowie die Adaptierbarkeit des entwickelten aufgaben-orientierten Programmiersystems gezeigt werden.

8 Technische und wirtschaftliche Bewertung

8.1 Methodische und technische Bewertung

Das entwickelte Informationsmodell, das darauf aufbauende adaptierbare Programmiersystem und das entwickelte Vorgehensmodell bieten die im Folgenden beschriebenen Vorteile.

Wie bei jedem aufgabenorientierten Programmiersystem ist ein geringeres Vorwissen des Anwenders notwendig und die Programmierung kann z. B. auch durch Werker erfolgen. Dieser kann sich damit auf die eigentlichen Prozesse konzentrieren. Insbesondere zusammen mit einer automatischen Konfiguration des Systems ist somit eine schnelle Umstellung eines Montagesystems auf neue Produkte oder Varianten möglich.

Den Anforderungen entsprechend, wurde ein Vorgehensmodell als Basis zur Integration der aufgabenorientierten Programmierung ausgewählt. Dieses wurde um die spezifischen Inhalte für die aufgabenorientierte Programmierung erweitert und erfüllt somit die gestellten methodischen Anforderungen. Es ermöglicht die einfache Abschätzung der Einsetzbarkeit der aufgabenorientierten Programmierung für ein Unternehmen und unterstützt den Anwender bei der Integration, indem die wesentlichen Schritte und Aufgaben dargelegt werden.

Die entwickelte Blackboard-Architektur erfüllt die strukturellen Anforderungen bzgl. der Anpassungsfähigkeit der Aufgabenkomplexität, der abgebildeten Prozesse und der Eingabesysteme. Neue Prozesse und Funktionalitäten können einfach über Wissensquellen integriert werden, ohne das Planungsmodul strukturell zu ändern. Die Anbindung von Simulationswerkzeugen wurde darüber hinaus in der Umsetzung gezeigt. Das Informationsmodell folgt der geforderten Modellierungsstruktur. Die darin enthaltenen Skills ermöglichen eine Unabhängigkeit von den Spezifika einzelner Hersteller. Mit dem Vorgehen zur Identifikation von direkten Skills konnten auch die speziell an die Skills gestellten Anforderungen erfüllt werden. Da das Vorgehen darüber hinaus auf eine Abgrenzung durch quantifizierbare Größen aufgebaut ist, kann es auch als Grundlage für Vorhaben zur Standardisierung dienen. Die vorgeschlagene Modellierung der Skills ermöglicht über Pointer ein Mapping auf spezifische Ressourcen.

Der Einsatz der entwickelten Konzepte ist teilweise limitiert, was nachstehend diskutiert wird. Bei der Betrachtung aktueller Entwicklungsprozesse ist für den

Einsatz des Programmiersystems noch zusätzlicher Modellierungsaufwand beim Anwender oder bei den Herstellern von Montagesystemkomponenten notwendig. Da jedoch davon auszugehen ist, dass die Digitalisierung weiter zunimmt, wird diese Einschränkung in Zukunft an Relevanz verlieren. Die entwickelten Konzepte umfassen zudem noch nicht alle Bestandteile eines Steuerungsprogramms und ein Automatismus zum Abgleich der Modelle mit der Realität, z. B. über Sensoren, fehlt. Insbesondere die Anbindung an ein MES stellt eine wichtige Weiterentwicklung dar. Für den erfolgreichen Einsatz des aufgabenorientierten Programmiersystems sind auch weiterhin Experten notwendig, die Erweiterungen des Systems durchführen können.

Im Rahmen dieser Arbeit war es nicht möglich eine komplette Evaluierung aller Inhalte der entwickelten Konzepte durchzuführen. Im Folgenden sollen jedoch Aussagen zur Gültigkeit der Ergebnisse dieser Arbeit aus der prototypischen Erprobung abgeleitet werden. In der Erprobung wurden mit den Montageprozessen „Schweißen“, „Handhaben“, „Prüfen“ und „Fördern“ sowie der Sonderoperation „Drucken“ Vertreter aus fast alle Bereichen der in Abbildung 3 dargestellten Montageprozesse ausgewählt. Nur für den Bereich der Justage erfolgte keine Erprobung. Die grundsätzliche Eignung des Informationsmodells sowie des adaptierbaren Planungsmoduls konnten damit gezeigt werden. Durch die schnelle und einfache Anpassung des Grundaufbaus an die Referenzszenarien konnten insbesondere die Vorteile des adaptierbaren Grundaufbaus des Programmiersystems gezeigt werden. Da außerdem die Prozesse des Klebens und Lötens in ihren zu modellierenden Elementen ähnlich einem Schweißprozess sind, kann auf eine einfache Übertragbarkeit geschlossen werden. Die Grenzen der Modellierungskonzepte sind aktuell insbesondere bei der Modellierung von formlabilen Objekten und Flüssigkeiten zu sehen. Eine Integration der notwendigen Informationen in das Modell des Produktes ist zwar vorgesehen und möglich, wurde jedoch nicht erprobt.

8.2 Wirtschaftliche Bewertung

Engineeringprozesse unterscheiden sich von Unternehmen zu Unternehmen. Damit kann auch bei der Wirtschaftlichkeitsbewertung nicht von einem allgemeingültigen Fall ausgegangen werden. Die Wirtschaftlichkeit des Einsatzes des aufgabenorientierten Programmiersystems wird im Wesentlichen von drei variablen Einflussgrößen bestimmt. Dies sind die evtl. notwendigen Aufwände, um alle benötigten Eingangsdaten wie z. B. Prozessbeschreibungen zu erzeugen, die

Häufigkeit der Umprogrammierung bzw. der insgesamt anfallende Aufwand zur Programmierung der Montagesysteme sowie die prozentuale Zeitersparnis bei der Programmierung. Die prozentuale Zeitersparnis ist von der verwendeten Ebene im Aufgabenmodell abhängig, da von dieser auch die notwendigen Nutzereingaben abhängen. So kann sich die Einführung auch bei hohen Investitionskosten durch häufige Umprogrammierung in kurzer Zeit amortisieren. Aus dieser Betrachtung können die wirtschaftlichen Grenzen für den Einsatz des aufgabenorientierten Programmiersystems abgeleitet werden. Im Folgenden wird zunächst eine beispielhafte Wirtschaftlichkeitsbetrachtung nach der Kapitalwertmethode dargestellt, um anschließend eine Grenzwertbetrachtung durchzuführen. Die Tabelle 5 zeigt die angenommenen Kenngrößen für die Wirtschaftlichkeitsbetrachtung und Tabelle 6 deren Ergebnis. Die angenommenen Zahlen repräsentieren ein fiktives Unternehmen mit 500 Mitarbeitern, welches auf die Herstellung von Montagesystemen spezialisiert ist. Die angenommene prozentuale Einsparung orientiert sich an den Ergebnissen der Erprobung. Basierend auf typischen Lizenzkosten kommerzieller Simulationssysteme wurden die Investitionskosten für das Programmiersystem festgelegt.

Tabelle 5: Angenommene Kennzahlen für die beispielhafte Wirtschaftlichkeitsrechnung

Unternehmenskennzahlen			
Abschreibungszeitraum	AB	5	Jahre
Kalk. Zinsen	Z	10	%
Wartungskosten	W	15	%
Stundensatz MA	S_{MA}	50	€

Investition			
Invest Grundaufbau aufgabenorientiertes Programmiersystem		7.000	€
Anpassung aufgabenorientiertes Programmiersystem (HMI, Wissensquellen, etc.)		10.000	€
Organisatorische Aufwände zur Integration		7.000	€
Summe Invest	**I**	24.000	€

Jährliche Einsparungen durch aufgabenorientiertes Programmiersystem			
Stunden für klassiche Programierung	t_{prog}	1.000	h/ Jahr
Einsparung durch aufgabenorientierte Programmierung	d	45	%
Summe jährliche Einsparungen	**$E=(t_{prog} * S_{MA}) * d$**	22.500	h/ Jahr
Jährliche Kosten durch aufgabenorientiertes Programmiersystem			
Wartung jährlich	$W_J = Z * I$	3.600	€/Jahr
Zusätzlicher Aufwand im Engineering	$A_{eng} = 100\ h * S_{MA}$	5.000	€/Jahr
Summe jährliche Kosten	**$K=A_{eng} + W_J$**	8.600	€/Jahr

Tabelle 6: Ergebnisse der beispielhaften Wirtschaftlichkeitsrechnung

Aufgabenorientierte Programmierung	Jahr	Einnahmen	Ausgaben	Überschuss	Kapitalwertmethode Abzinsfaktor AF	Barwert der Überschüsse	Kapitalwert
	0		24.000,00 €	-24.000,00 €	1,000000	-24.000,00 €	-24.000,00 €
	1	22.500,00 €	8.600,00 €	13.900,00 €	0,909091	12.636,36 €	-11.363,64 €
	2	22.500,00 €	8.600,00 €	13.900,00 €	0,826446	11.487,60 €	123,97 €
	3	22.500,00 €	8.600,00 €	13.900,00 €	0,751315	10.443,28 €	10.567,24 €
	4	22.500,00 €	8.600,00 €	13.900,00 €	0,683013	9.493,89 €	20.061,13 €
	5	22.500,00 €	8.600,00 €	13.900,00 €	0,620921	8.630,81 €	28.691,94 €
Summe		**112.500,00 €**	**67.000,00 €**	**45.500,00 €**		**52.691,94 €**	**28.691,94 €**

Barwert der Überschüsse=	**52.691,94 €**
Anschaffungskosten=	24.000,00 €
Kapitalwert$_{ges}$=	**28.691,94 €**

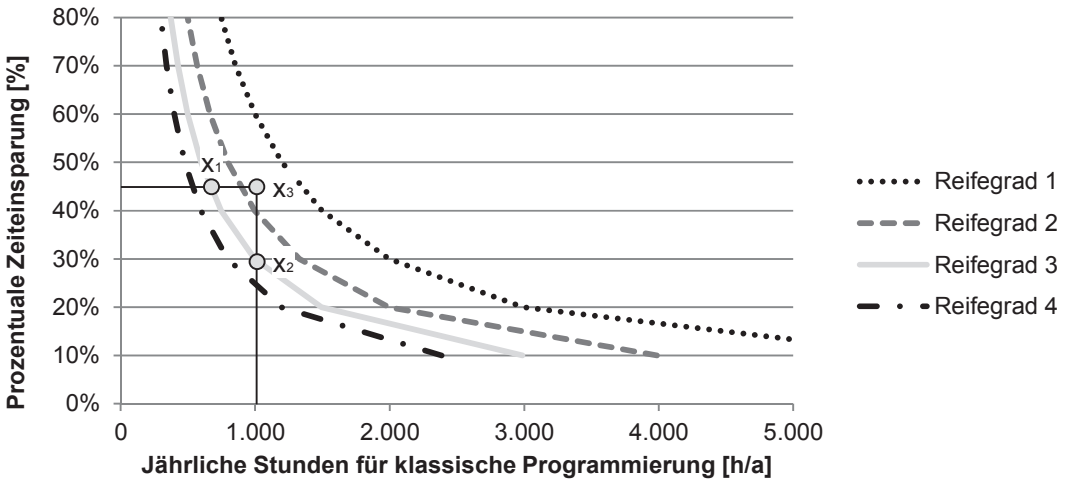
Abzinsfaktor(Jahr) = $(1+\text{Zinssatz})^{-\text{Jahr}}$

Barwert(Jahr) = Überschuss(Jahr) * Abzinsfaktor(Jahr)

Kapitalwert(Jahr) = Kapitalwert(Jahr-1) + Barwert der Überschüsse(Jahr)

Kaptialwert_{ges} = Kaptialwert(Jahr 1) + Kaptialwert(Jahr 2) + Kaptialwert(Jahr

Für die Grenzwertbetrachtung wurde ein Vorgehen angewendet, das an VOGL (2009) angelehnt ist. Dazu wird der zusätzliche Aufwand im Engineering zunächst als konstant angenommen und die prozentuale Zeiteinsparung und die jährlichen Stunden für die klassische Programmierung als Variablen verwendet. Für diese können nun die Isolinien eingetragen werden (vgl. Reifegrad 3 in Abbildung 56). Eine Isolinie entspricht Wertepaarungen von prozentualer Zeiteinsparung und der jährlichen Stunden für die klassische Programmierung, die mit den oben verwendeten Kennzahlen und Investitionen auf den Abschreibungszeitraum den Kapitalwert Null ergeben (vgl. Punkte x_1 und x_2). Werte oberhalb der Isolinie gelten als vorteilhafte Investitionen (vgl. Punkt x_3 welcher dem oben beschriebenen Szenario entspricht). Darunter liegende Wertepaare sind nicht vorteilhaft. Niedrigere Reifegrade eines Unternehmens führen bei ansonsten gleichbleibenden Kennzahlen zu höheren laufenden Kosten im Engineering. Diese können somit als weitere Isolinien in die Übersicht eingetragen werden (vgl. Abbildung 56). Aus diesen ist ersichtlich, dass mit sinkendem Reifegrad höhere Anteile der Zeitersparnis oder der Stunden für die Programmierung notwendig sind, um eine wirtschaftlich vorteilhafte Investition in das aufgabenorientierte Programmiersystem tätigen zu können.



Reifegrad 1 ≡ 20.000 € zusätzlicher Aufwand Engineering, Reifegrad 2 ≡ 10.000 € zusätzlicher Aufwand Engineering
Reifegrad 3 ≡ 5.000 € zusätzlicher Aufwand Engineering; Reifegrad 4 ≡ 2.000 € zusätzlicher Aufwand Engineering

Abbildung 56: Grenzwertbetrachtung der Wirtschaftlichkeit

Somit kann zusammenfassend festgestellt werden, dass Unternehmen einen höheren Reifegrad in ihren Unternehmens- und Engineeringprozessen anstreben sollten, um das Konzept der aufgabenorientierten Programmierung effizienzsteigernd einsetzten zu können. Insbesondere die verfügbaren digitalen Daten sind dabei wesentliche Voraussetzungen. Sind diese erfüllt, ist der effiziente Einsatz der aufgabenorientierten Programmierung auch bei bisher nur geringen Aufwendungen für Programmiertätigkeiten möglich.

9 Zusammenfassung und Ausblick

Aus den aktuell vorherrschenden globalen Randbedingungen ergibt sich die Anforderung, die Programmierung von Montagesystemen zu vereinfachen. Der Ansatz der aufgabenorientierten Programmierung stellt eine Möglichkeit dar, diese Anforderung zu erfüllen. Sogenannte Skills ermöglichen die Verknüpfung von abstrakten Ablaufbeschreibungen mit den Funktionalitäten von Ressourcen im Montagesystem. Im Kapitel „Stand der Wissenschaft und Technik“ wurde dargelegt, dass schon eine Vielzahl an Ansätzen an dieses Konzept anknüpft. Diese sind jedoch noch mit Einschränkungen verbunden. So werden u. a. die Skills nicht hergeleitet, der Fokus liegt nur auf Robotersteuerungen oder es wird bei der Aufgabe von einem festen Abstraktionsgrad ausgegangen.

Daher wurde im Rahmen dieser Arbeit ein adaptierbares, aufgabenorientiertes Programmiersystem entwickelt. Dieses baut auf einem Skill-basierten Informationsmodell auf, welches den Schwerpunkt dieser Arbeit darstellt. Um einzelne Skills voneinander abgrenzen zu können, wurde ein zugehöriges Vorgehen entwickelt und angewendet. Die Modellierung der Skills umfasste insbesondere ein Mapping auf die Ressourcen und deren Ansteuerungsmöglichkeiten durch sogenannte Pointer. Dies macht es möglich, eine adaptierbare Struktur des Programmiersystems zu entwerfen, welches durch die Abstraktion der Skills unabhängig von den besonderen Eigenheiten einzelner Ressourcen bzw. Hersteller ist. Ein hierarchisches Aufgabenmodell ermöglicht eine flexible Wahl der Komplexität der Aufgabenbeschreibung. Um Unternehmen bei der Einführung eines aufgabenorientierten Programmiersystems zu unterstützen, wurde außerdem ein Vorgehensmodell abgeleitet. Durch das dort enthaltene Reifegradmodell können Unternehmen aufwandsarm abschätzen, ob der Einsatz eines aufgabenorientierten Programmiersystems für sie vorteilhaft ist. Das Konzept der Modellierung und das des adaptierbaren, aufgabenorientierten Programmiersystems wurden anhand einer Roboterzelle und eines SPS-gesteuerten Montagemoduls umgesetzt und erprobt. Weitere Anwendungsfälle konnten darüber hinaus in der Simulation gezeigt werden. Abschließend erfolgte eine technische und wirtschaftliche Bewertung der erarbeiteten Ergebnisse.

Für die im Rahmen dieser Arbeit entwickelten Ergebnisse sind jedoch zusätzliche Weiterentwicklungen möglich, die im Folgenden skizziert werden. Das Konzept der Skills könnte aufgegriffen und für weitere Anwendungsfälle, z. B. aus dem Bereich der Verarbeitungsmaschinen verwendet werden. Aus den Wirkpaarungen der Verarbeitungsprozesse lassen sich einzelne Funktionen ableiten, welche z. B. von STICH ET AL. (2013) beschrieben werden. Diese könnten als Grundlage der Skills

für Verarbeitungsmaschinen dienen. Die Modellierung und Unterteilung der Skills kann darüber hinaus auch für unternehmensinterne Bibliotheken verwendet werden. Hier könnte das Skill-Konzept integriert werden, um eine größere Flexibilität zu erreichen.

Neben der Modellierung können darüber hinaus das Programmiersystem und dessen Architektur für einen Onlinebetrieb oder hybriden Betrieb erweitert werden. Damit könnten online geänderte Programmanweisungen generiert werden, wenn etwa eine neue Variante montiert werden soll. Die Blackboard-Architektur ist prinzipiell dafür geeignet, bei einer Umsetzung ist jedoch auf besondere Effizienz der Wissensquellen zu achten. Nur dann können die Planungsergebnisse in kurzer Zeit zur Verfügung gestellt werden. Grundsätzlich stellt auch die Integration weiterer Wissensquellen auf Spezialistenebene eine Möglichkeit der Erweiterung dar, welche ohne Einschränkung erfolgen kann. Außerdem ist eine Weiterverwendung der zur Programmierung verwendeten Modelle zielführen. Ein Beispiel dafür wäre eine betriebsparallele Simulation der Anlage bzw. des Montagesystems, aus der etwa auf fehlerhafte Zustände geschlossen werden kann. Ein entsprechendes Konzept wird beispielsweise von KAIN ET AL. (2008) vorgestellt.

10 Literaturverzeichnis

3S-SMART 2014
3S-Smart Software Solutions (Hrsg.): CODESYS Engineering - Plattform für die industrielle Automatisierung. <http://de.codesys.com/produkte/codesys-engineering.html> - 17.02.2015.

ABB ROBOTICS 2015
ABB Robotics (Hrsg.): RobotStudio.
<http://new.abb.com/products/robotics/robotstudio> - 17.02.2015.

ABELE & REINHART 2011
Abele, E.; Reinhart, G.: Zukunft der Produktion. Herausforderungen, Forschungsfelder, Chancen. München: Hanser, Carl 2011. ISBN: 3446425950.

AHLEMANN ET AL. 2005
Ahlemann, F.; Schröder, C.; Teuteberg, F.: Kompetenz- und Reifegradmodelle für das Projektmanagement. Grundlagen, Vergleich und Einsatz. Osnabrück: ISPRI 2005. ISBN: 9783936475241. (ISPRI-Arbeitsbericht, Nr. 2005, 01).

ANSI/ISA 88
ANSI/ISA 88: Batch Control: 2010.

ANSI/ISA 95
ANSI/ISA 95: Enterprise-Control System Integration: 2000.

ARAI ET AL. 2000
Arai, T.; Aiyama, Y.; Maeda, Y.; Sugi, M.; Ota, J.: Agile Assembly System by "Plug and Produce". CIRP Annals - Manufacturing Technology 49 (2000) 1, S. 1-4.

ARAI ET AL. 2003
Arai, T.; Izawa, H.; Maeda, Y.; Kikuchi, H.; Ogawa, H.; Sugi, M.: Real-time task decomposition and allocation for a multi-agent robotic assembly cell. In: IEEE (Hrsg.): 5th IEEE International Symposium on Assembly and Task Planning (ISATP'03). Besancon, France, 10.-11. Juli 2003, S. 42-47.

ARAMAKI ET AL. 1999
Aramaki, S.; Nagasawa, I.; Kurono, S.; Nagai, T.; Morita, T.; Suetsugu, T.: The knowledge representation and programming for robotic assembly task. In: IEEE (Hrsg.): 1999 IEEE International Symposium on Assembly and Task Planning (ISATP'99). Porto, Portugal, 21.-24. Juli 1999, S. 256-261.

ARAMAKI ET AL. 2007
Aramaki, S.; Nagai, T.; Kawamura, M.; Yayoshi, K.; Hatada, Y.; Tsuruoka, T.: A Robot Programming Based on Frame Representation of Knowledge. In: IEEE (Hrsg.): 7th IEEE International Conference on Computer and Information Technology (CIT 2007). Aizu-Wakamatsu, Fukushima, Japan, 16.-18. Oktober 2007, S. 903-908.

ARMBRUSTER ET AL. 2009
Armbruster, H.; Kirner, E.; Kinkel, S.: Neue Nutzungspotentiale für Industrieroboter. wt Werkstattstechnik 96 (2009) 9, S. 631-636.

B&R 2015
Bernecker + Rainer Industrie Elektronik GmbH (Hrsg.): B&R: Automation Studio. <http://www.br-automation.com/de-de/produkte/software/automation-studio/> - 24.06.2015.

BACKHAUS & REINHART 2013
Backhaus, J.; Reinhart, G.: Efficient application of task-oriented programming for assembly systems. In: IEEE (Hrsg.): 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). Wollongong, NSW, Australien, 9.-12. Juli 2013, S. 750-755.

BACKHAUS & REINHART 2014
Backhaus, J.; Reinhart, G.: Adaptive and Device Independent Planning Module for task-oriented Programming of Assembly Systems. In: CIRP (Hrsg.): 9th CIRP Conference on Intelligent Computation in Manufacturing Engineering (ICME). Neapel, Italien, 23.-25. Juli 2014.

BACKHAUS & REINHART 2015
Backhaus, J.; Reinhart, G.: Digital Description of Products, Processes and Resources for task-oriented Programming of Assembly Systems. Journal of Intelligent Manufacturing (2015).

BACKHAUS ET AL. 2011
Backhaus, K.; Erichson, B.; Plinke, W.; Weiber, R.: Multivariate Analysemethoden. Eine anwendungsorientierte Einführung. 13., überarb. Aufl. Berlin [u.a.]: Springer 2011. ISBN: 3642164919. (Springer-Lehrbuch).

BACKHAUS ET AL. 2013
Backhaus, J.; Ulrich, M.; Reinhart, G.: Classification, Modelling and Mapping of Skills in Automated Production Systems. In: Zaeh, M. F. (Hrsg.): Enabling Manufacturing Competitiveness and Economic Sustainability - Proceedings of the

5th International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2013). München, 6.-9. Oktober 2013. Cham: Springer International Publishing 2013, S. 85-89. ISBN: 978-3-319-02053-2.

BALZERT 2011
Balzert, H.: Lehrbuch der Software-Technik. Entwurf, Implementierung, Installation und Betrieb. 3. Aufl. Heidelberg [u.a.]: Spektrum, Akad. Verl. 2011. ISBN: 3827422469. (Lehrbücher der Informatik).

BARBIERI ET AL. 2013
Barbieri, G.; Fantuzzi, C.; Borsari, R.: Tools for the development of a design methodology for mechatronic systems. (Hrsg.): IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA). Cagliari, Italy, 10.-13. September 2013, S. 1-4.

BAUERNHANSL 2014
Bauernhansl, T.: Die Vierte Industrielle Revolution – Der Weg in ein wertschaffendes Produktionsparadigma. In: Bauernhansl, T. et al. (Hrsg.): Industrie 4.0 in Produktion, Automatisierung und Logistik. Anwendung, Technologien, Migration. Dordrecht: Springer 2014, S. 5-35. ISBN: 978-3-658-04682-8.

BECKHOFF AUTOMATION 2015
Beckhoff Automation GmbH & Co. KG (Hrsg.): TwinCAT 3. <http://www.beckhoff.de/german/twincat/twincat-3.htm> - 17.02.2015.

BENDER ET AL. 2005
Bender, K.; Dominka, S.; Koc, A.; Pöschl, M.; Russ, M.; Stützel, B.: Embedded Systems - qualitätsorientierte Entwickung. 1. Aufl. Berlin: Springer 2005. ISBN: 9783540229957.

BENGEL 2007
Bengel, M.: Modelling objects for skill-based reconfigurable machines. In: Pham, D. T. et al. (Hrsg.): Innovative production machines and systems - Third I*PROMS Virtual Conference. Dunbeath, Schottland, 2.-13. Juli 2007. Dunbeath: Whittles 2008, S. 238-243. ISBN: 978-1-904445-52-4.

BENGEL 2010
Bengel, M.: Workpiece-centered Approach to Reconfiguration in Manufacturing Engineering. Diss. Universität Stuttgart 2010.

BERGERT ET AL. 2007
Bergert, M.; Diedrich, C.; Kiefer, J.; Bar, T.: Automated PLC software generation based on standardized digital process information. In: IEEE (Hrsg.): IEEE

Conference on Emerging Technologies & Factory Automation (EFTA 2007). Patras, Griechenland, 25.-28. Sept. 2007, S. 352-359.

BERGERT ET AL. 2010
Bergert, M.; Höme, S.; Hundt, L.: Verhaltensmodellierung für die Virtuelle Inbetriebnahme. etz - Elektrotecnik & Automation 9 (2010), S. 2-9.

BERNHARDT ET AL. 1995
Bernhardt, R.; Schreck, G.; Willnow, C.: Realistic robot simulation. Computing & Control Engineering Journal 6 (1995) 4, S. 174-176.

BI & LANG 2007
Bi, Z. M.; Lang, S.: Automated robotic programming for products with changes. International Journal of Production Research 45 (2007) 9, S. 2105-2118.

BIEGELBAUER ET AL. 2004
Biegelbauer, G.; Vincze, M.; Nohmayer, H.; Eberst, C.: Sensor based robotics for fully automated inspection of bores at low volume high variant parts. In: IEEE (Hrsg.): Proceedings International Conference on Robotics and Automation (ICRA). New Orleans, LA, USA, 26. April - 1. Mai 2004, S. 4852-4857.

BJÖRKELUND ET AL. 2011
Björkelund, A.; Malec, J.; Nilsson, K.; Nugues, P.: Knowledge and Skill Representations for Robotized Production. In: Bittanti, S. (Hrsg.): Proceedings of the 18th IFAC World Congress. Mailand, Italien, 28. August - 2. September. New York: IFAC 2011. ISBN: 3902661933.

BLACKBIRD 2015
Blackbird GmbH (Hrsg.): RobotMotionCenter. <http://www.blackbird-robotics.de/produkte-loesungen/robotmotioncenter.html> - 17.02.2015.

BLEY & BOSSMANN 2005
Bley, H.; Bossmann, M.: Standardisierte Produktmodelle für die automatisierte Montageplanung. wt Werkstattstechnik 95 (2005) 9, S. 627-631.

BLEY & FRANKE 2005
Bley, H.; Franke, C.: Integration of Product Design and Assmbly Planning in the Digital Factory. Annals of the CIRP 53 (2005) 1, S. 25-30.

BONIN ET AL. 2014
Bonin, D.; Wischniewski, S.; Wirsching, H.-J.; Upmann, A.; Rausch, J.; Paul, G.: Exchanging data between Digital Human Modelling systems: a review of data formats. In: AIST (Hrsg.): 3rd International Digital Human Modeling Symposium. Tokyo, Japan, 20.-22. Mai 2014, S. 1-8.

BOSCH REXROTH 2015
Bosch Rexroth AG (Hrsg.): Software-Tools - IndraWorks Engineering. <http://www.boschrexroth.com/dcc/Vornavigation/VorNavi.cfm?PageID=p307594&Language=de> - 17.02.2015.

BOSSMANN 2007
Bossmann, M.: Feature-basierte Produkt- und Prozessmodelle in der integrierten Produktentstehung. [Online-Ausg.] Aufl. Saarbrücken: LFT, Univ. 2007. ISBN: 9783930429677. (Schriftenreihe Produktionstechnik, Nr. Bd. 38).

BRACHT ET AL. 2009
Bracht, U.; Wenzel, S.; Geckler, D.: Digitale Fabrik. Methoden und Praxisbeispiele. 1. Aufl. Berlin: Springer Berlin 2009. ISBN: 3540889736. (VDI-Buch).

BRECHER ET AL. 2004
Brecher, C.; Schröter, B.; Almeida, C.; Dai, F.; Matthias, B.; Kock, S.: Intuitiv bedienbare Programmiersysteme zur effizienten Programmierung von Handhabungsaufgaben. (Hrsg.): Robotik 2004: Leistungsstand - Anwendungen - Visionen - Trends. Düsseldorf 2004, S. 303-310.

BULLINGER 1986
Bullinger, H. J. (Hrsg.): Systematische Montageplanung. Handbuch fur die Praxis. Munich: Hanser 1986. ISBN: 978-3-446-14606-8.

BÜSCHER ET AL. 2013
Büscher, C.; Kuz, S.; Ewert, D.; Schilberg, D.; Jeschke, S.: Kognitive Planungs- und Lernmechanismen in selbstoptimierenden Montagesystemen. In: Jeschke, S. et al. (Hrsg.): Automation, Communication and Cybernetics in Science and Engineering 2011/2012. Berlin, Heidelberg: Springer 2013, S. 595-605. ISBN: 978-3-642-33388-0.

BUSCHMANN 2001
Buschmann, F.: A system of patterns. Chichester [u.a.]: Wiley 2001. ISBN: 978-0471958697. (Pattern-oriented software architecture, Nr. 1).

CÂNDIDO & BARATA 2007
Cândido, G.; Barata, J.: A Multiagent Control System for Shop Floor Assembly. In: Mařík, V. (Hrsg.): Holonic and multi-agent systems for manufacturing. Regensburg, 3.-5. September. Berlin, Heidelberg, New York: Springer 2007, S. 293-302. ISBN: 978-3-540-74478-8.

CAVIN ET AL. 2013
Cavin, S.; Ferreira, P.; Lohse, N.: Dynamic skill allocation methodology for

evolvable assembly systems. In: IEEE (Hrsg.): 2013 IEEE 11th International Conference on Industrial Informatics (INDIN). Bochum, Deutschland, 29.-31. Juli 2013, S. 218-223.

CHEN & SHENG 2011
Chen, H.; Sheng, W.: Transformative industrial robot programming in surface manufacturing. In: IEEE (Hrsg.): IEEE International Conference on Robotics and Automation (ICRA). Shanghai, China, 9.-13. Mai 2011, S. 6059-6064.

CLAUS & SCHWILL 2003
Claus, V.; Schwill, A.: Duden, Informatik. Ein Fachlexikon für Studium und Praxis. [Taschenbuchausg.] Aufl. Mannheim, Leipzig, Wien, Zürich: Dudenverlag 2003. ISBN: 9783411100231.

CMA ROBOTICS 2013
CMA Robotics (Hrsg.): Off-Line Painting Program – System der Off-line Programmerzeugung auf PC.
<http://www.cmarobot.it/lackierroboter/robot_software3.php> - 17.02.2015.

CONVERGENT 2015
Convergent Information Technologies GmbH (Hrsg.): AutomAPPPS.
<http://www.convergent-it.at/drupal/content/products> - 17.02.2015.

CORMEN ET AL. 2013
Cormen, T. H.; Leierson, C. E.; Rivest, R.; Stein, C.: Algorithmen - Eine Einführung. 4., durchges. und korrigierte Aufl. München: Oldenbourg 2013. ISBN: 978-3-486-74861-1.

CUIPER 2000
Cuiper, R.: Durchgängige rechnergestützte Planung und Steuerung von automatisierten Montagevorgängen. Diss. Technische Universität München. München 2000.

DAMMERTZ 1996
Dammertz, R.: Ein Programmiersystem zur graphisch strukturierten Erstellung von Roboterprogrammen und Programmieroberflächen. Als Ms. gedr Aufl. Aachen: Shaker 1996. ISBN: 9783826520266. (Berichte aus der Produktionstechnik, Nr. 30).

DELLIGATTI 2014
Delligatti, L.: SysML distilled. A brief guide to the systems modeling language. Upper Saddle River, NJ: Addison-Wesley 2014. ISBN: 978-0-321-92786-6.

DEMOLY ET AL. 2011
Demoly, F.; Yan, X.-T.; Eynard, B.; Rivest, L.; Gomes, S.: An assembly oriented design framework for product structure engineering and assembly sequence planning. Robotics and Computer-Integrated Manufacturing 27 (2011) 1, S. 33-46.

DENKENA ET AL. 2005
Denkena, B.; Wörn, H.; Apitz, R.; Bischoff, R.; Hein, B.; Kowalski, P.; Mages, D.; Schuler, H.: Roboterprogrammierung in der Fertigung. wt Werkstattstechnik 9.

DI ORIO ET AL. 2013
Di Orio, G.; Barata, J.; Sousa, C.; Flores, L.: Control System Software Design Methodology for Automotive Industry. In: IEEE (Hrsg.): IEEE International Conference on Systems, Man and Cybernetics (SMC 2013). Manchester, England, 13.-16 Oktober 2013 2013, S. 3848-3853.

DIEDRICH ET AL. 2005
Diedrich, D.; Franz, G.; John, K. H.; Krause, J.; Poignee, F.: Support of control application design using digital design and planing of manufacturing cells. In: Zítek, P. (Hrsg.): Proceedings of the 16th IFAC World Congress. Prag, Tschechien, 4. Juli: IFAC, Elsevier 2005, S. 1459.

DILLMANN & HUCK 1991
Dillmann, R.; Huck, M.: Informationsverarbeitung in der Robotik. Berlin: Springer 1991. ISBN: 3540530363.

DIN 4002-1
DIN 4002-1: Merkmale und Geltungsbereiche zum Produktdatenaustausch. Berlin: Beuth Verlag 2007.

DIN 6789-2
DIN 6789-2: Dokumentationssystematik - Verfälschungssicherheit und Qualitätskriterien für die Freigabe digitaler Produktdaten. Berlin: Beuth Verlag 2013.

DIN 8593
DIN 8593: Fertigungsverfahren Fügen. Berlin: Beuth Verlag 2003.

DIN EN 1560
DIN EN 1560: Gießereiwesen - Bezeichnungssystem für Gusseisen - Werkstoffkurzzeichen und Werkstoffnummern. Berlin: Beuth Verlag 2011.

DIN EN 573-1
DIN EN 573-1: Aluminium und Aluminiumlegierungen - Chemische

Zusammensetzung und Form von Halbzeug - Teil 1: Numerisches Bezeichnungssystem. Berlin: Beuth Verlag 2005.

DIN EN 61131
DIN EN 61131: Speicherprogrammierbare Steuerungen: Beuth Verlag 2014.

DIN EN 61499
DIN EN 61499: Funktionsbausteine für industrielle Leitsysteme. Berlin: Beuth Verlag 2014.

DIN EN 62264
DIN EN 62264: Integration von Unternehmensführungs- und Leitsystemen. Berlin: Beuth Verlag 2014.

DIN EN 62424
DIN EN 62424: Darstellung von Aufgaben der Prozessleittechnik - Fließbilder und Datenaustausch zwischen EDV-Werkzeugen zur Fließbilderstellung und CAE-Systemen. Berlin: Beuth Verlag 2010.

DIN EN 62714
DIN EN 62714: Datenaustauschformat für Planungsdaten industrieller Automatisierungssysteme. Berlin: Beuth Verlag 2013.

DIN EN ISO 1043
DIN EN ISO 1043: Kunststoffe - Kennbuchstaben und Kurzzeichen. Berlin: Beuth Verlag 2012.

DIN EN ISO 8373
DIN EN ISO 8373: Roboter und Robotikgeräte - Wörterbuch. Berlin: Beuth Verlag 2010.

DIN IEC 60050-351
DIN IEC 60050-351: Internationales Elektrotechnisches Wörterbuch. Berlin: Beuth Verlag 2014.

DIN ISO 2768-1
DIN ISO 2768-1: Allgemeintoleranzen; Toleranzen für Längen- und Winkelmaße ohne einzelne Toleranzeintragung. Berlin: Beuth Verlag 1991.

DOMBROWSKI & SCHMIDTCHEN 2010
Dombrowski, U.; Schmidtchen, K.: Ganzheitliche Produktionssysteme. ZWF - Zeitschrift für wirtschaftlichen Fabrikbetrieb 105 (2010) 10, S. 914-918.

DOMINKA 2007
Dominka, S.: Hybride Inbetriebnahme von Produktionsanlagen - von der virtuellen

zur realen Inbetriebnahme. 1. Aufl. Göttingen: Sierke 2007. ISBN: 3940333409. (Informationstechnik im Maschinenwesen).

DRATH & FEDAI 2004A
Drath, R.; Fedai, M.: CAEX – ein neutrales Datenaustauschformat für Anlagendaten. Teil 1. atp – Automatisierungstechnische Praxis 46 (2004a) 2, S. 52-56.

DRATH & FEDAI 2004B
Drath, R.; Fedai, M.: CAEX – ein neutrales Datenaustauschformat für Anlagendaten. Teil 2. atp – Automatisierungstechnische Praxis 46 (2004b) 3, S. 20-27.

DRATH 2010
Drath, R.: Datenaustausch in der Anlagenplanung mit AutomationML. Integration von CAEX, PLCopen XML und COLLADA. Heidelberg, New York: Springer 2010. ISBN: 978-3642046735.

DRESCHER ET AL. 2013
Drescher, B.; Stich, P.; Kiefer, J.; Strahilov, A.; Bär, T.; Reinhart, G.: Physikbasierte Simulation im Anlagenentstehungsprozess - Einsatzpotenziale bei der Entwicklung automatisierter Montageanlagen im Automobilbau. In: Dangelmaier, W. et al. (Hrsg.): Simulation in Produktion und Logistik 2013. Paderborn, 09. - 11. Oktober 2013. Paderborn: Heinz-Nixdorf-Inst. Univ. Paderborn 2013, S. 270-281. ISBN: 9783942647359.

DRÖSCHEL ET AL. 1997
Dröschel, W.; Heuser, W.; Midderhoff, R.: Inkrementelle und objektorientierte Vorgehensweisen mit dem V-Modell 97. München, Wien: Oldenbourg 1997. ISBN: 9783486242768.

DUDA ET AL. 2001
Duda, R. O.; Hart, P. E.; Stork, D. G.: Pattern classification. 2. Aufl. New York: Wiley 2001. ISBN: 9780471429777.

DYLA 2002
Dyla, A.: Modell einer durchgängig rechnerbasierten Produktentwicklung. 2002. ISBN: 3000096841. (FZG, Nr. 131).

EHRMANN 2007
Ehrmann, M.: Beitrag zur Effizienzsteigerung bei der Programmierung flexibler, roboterbasierter Montagezellen. Konzeption und Realisierung eines nutzergerechten Programmiersystems. Techn. Univ., Diss.--Kaiserslautern, 2007. Als Ms. gedr Aufl.

Kaiserslautern: Techn. Univ 2007. ISBN: 978-3-939432-38-8. (Fortschritt-Berichte pak Robotik, Nr. 15).

EIGNER & STELZER 2009
Eigner, M.; Stelzer, R.: Product Lifecycle Management. Ein Leitfaden für Product Development und Life Cycle Management. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg 2009. ISBN: 3540684018.

EIGNER ET AL. 2014
Eigner, M.; Roubanov, D.; Zafirov, R.: Modellbasierte virtuelle Produktentwicklung. Berlin, Heidelberg: Springer 2014. ISBN: 366243816X.

ENG ET AL. 1999
Eng, T.-H.; Ling, Z.-K.; Olson, W.; McLean, C.: Feature-based assembly modeling and sequence generation. Computers & Industrial Engineering 36 (1999) 1, S. 17-33.

EPPLE 2003
Epple, U.: Austausch von Anlagenplanungsdaten auf der Grundlage von Metamodellen. atp – Automatisierungstechnische Praxis 45 (2003) 7, S. S. 2–11 .

ESTER & SANDER 2000
Ester, M.; Sander, J.: Knowledge discovery in databases. Techniken und Anwendungen. Berlin, New York: Springer 2000. ISBN: 3-540-67328-8.

ESTÉVEZ ET AL. 2007
Estévez, E.; Marcos, M.; Orive, D.: Automatic generation of PLC automation projects from component-based models. The International Journal of Advanced manufacturing Technology 35 (2007) 5-6, S. 527-540.

EWERT ET AL. 2010
Ewert, D.; Thelen, S.; Kunze, R.; Mayer, M.; Schilberg, D.; Jeschke, S.: A Graph Based Hybrid Approach of Offline Pre-planning and Online Re-planning for Efficient Assembly under Realtime Constraints. In: Hutchison, D. et al. (Hrsg.): Third International Conference on Intelligent Robotics and Applications (ICIRA). Shanghai, China, 10.-12. November. Berlin, Heidelberg: Springer 2010, S. 44-55. ISBN: 978-3-642-16586-3.

FALKMAN ET AL. 2011
Falkman, P.; Helander, E.; Andersson, M.: Automatic generation: A way of ensuring PLC and HMI standards. In: IEEE (Hrsg.): IEEE 16th Conference on Factory Automation (ETFA 2011). Toulouse, France, 5.-9. Sept. 2011, S. 1-4.

FANDEL ET AL. 1994
Fandel, G.; Dyckhoff, H.; Reese, J.: Industrielle Produktionsentwicklung. Eine empirisch-deskriptive Analyse ausgewählter Branchen ; mit 44 Tabellen. 2. Aufl. Berlin [u.a.]: Springer 1994. ISBN: 3540578471.

FANUC AMERICA 2015
FANUC K.K. (Hrsg.): ROBOGUIDE - FANUC Simulation Software. <http://robot.fanucamerica.com/products/vision-software/ROBOGUIDE-simulation-software.aspx> - 23.06.2015.

FELDMANN 1997
Feldmann, C.: Eine Methode für die integrierte rechnergestützte Montageplanung. Diss. Diss.; Technische Universität München. Berlin, New York: Springer 1997. ISBN: 9783540620594. (Forschungsberichte IWB, Nr. 104).

FELDMANN ET AL. 2014
Feldmann, K.; Schöppner, V.; Spur, G. (Hrsg.): Handbuch Fügen, Handhaben, Montieren. 2., vollst. neu bearb. Aufl. München: Hanser 2014. ISBN: 9783446428270. (Edition Handbuch der Fertigungstechnik).

FISCHER 1981
Fischer, P.: Rechnerunterstützte Erstellung von Schaltplänen am Beispiel der automatischen Hydraulikplanzeichnung. Berlin, Heidelberg, New York: Springer 1981. ISBN: 9780387106427. (ISW, Nr. 35).

FLEISCHMANN 2010
Fleischmann, A.: What Is S-BPM? In: Buchwald, H. et al. (Hrsg.): S-BPM ONE – Setting the Stage for Subject-Oriented Business Process Management. Berlin, Heidelberg: Springer 2010, S. 85-106. ISBN: 978-3-642-15914-5.

FORSTNER & DÜMMLER 2014
Forstner, L.; Dümmler, M.: Integrierte Wertschöpfungsnetzwerke – Chancen und Potenziale durch Industrie 4.0. e & i Elektrotechnik und Informationstechnik 131 (2014) 7, S. 199-201.

FREESE ET AL. 2010
Freese, M.; Singh, S.; Ozaki, F.; Matsuhira, N.: Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator. In: Hutchison, D. et al. (Hrsg.): Simulation, Modeling, and Programming for Autonomous Robots. Darmstadt, 15.-18. November. Berlin, Heidelberg: Springer 2010, S. 51-62. ISBN: 978-3-642-17318-9.

FREI ET AL. 2009
Frei, R.; Ferreira, B.; Marzo Serugendo, G.; Barata, J.: An architecture for self-managing evolvable assembly systems. In: IEEE (Hrsg.): 2009 IEEE International Conference on Systems, Man and Cybernetics - SMC. San Antonio, TX, USA, 11.-14. Oktober 2009, S. 2707-2712.

FREUND ET AL. 1990
Freund, E.; Heck, H.; Kreft, K.; Mauve, C.: OSIRIS - Ein objektorientiertes System zur impliziten Roboterprogrammierung und Simulation. Robotersysteme : Zeitschrift für Informationstechnologie u. Handhabungstechnik (1990) 6, S. 185-192.

FRIEDEWALD ET AL. 2011
Fraunhofer IGD (Hrsg.): Benchmark neutraler Formate für den prozessübergreifenden Datenaustausch im Schiffbau. <http://publica.fraunhofer.de/documents/N-194622.html> - 08.02.2015.

FRIEDRICH 2010
Friedrich, T.: Technologieorientiertes Programmier- und Steuerungssystem für Industrieroboter. Diss. Techn. Universität Berlin 2010.

GALANTUCCI ET AL. 2004
Galantucci, L. M.; Percoco, G.; Spi, R.: Assembly and Disassembly Planning by using Fuzzy Logic & Genetic Algorithms. International Journal of Advanced Robotic Systems 1 (2004) 2, S. 67-74.

GANGHOFF 1993
Ganghoff, P.: Wissensbasierte Unterstützung der Planung technischer Systeme. Dissertation. Inst. für Werkzeugmaschinen und Betriebstechnik. Karlsruhe 1993.

GERKE 2014
Gerke, W.: Technische Assistenzsysteme: vom Industrieroboter zum Roboterassistenten: De Gruyter 2014. ISBN: 9783110343717.

GHOFFRANI 2008
Ghoffrani, M.: Entwicklung und Einführung eines flexiblen Softwaresystems zur Konfigurierung virtueller Produkte. Bochum, Univ., Diss.,. Aachen: Shaker 2008. ISBN: 9783832269524. (Maschinenbauinformatik, Nr. 2008,1).

GÖRING & FAY 2012
Göring, M.; Fay, A.: Modellierung von Veränderungen in hierarchischen Strukturmodellen automatisierter Anlagen. Softwaretechnik-Trends 32 (2012) 2, S. 80-81.

GOTTSCHALD 2001
Gottschald, J.: Place & play-Roboter. Ein portables Handhabungssystem für die Werkstatt. Aachen: Shaker 2001. ISBN: 9783826592751. (Berichte aus der Produktionstechnik, Nr. Bd. 2001,22).

GRUNWALD 2002
Grunwald, S.: Methode zur Anwendung der flexiblen integrierten Produktentwicklung und Montageplanung. Diss., Technische Universität München. München: Utz 2002. ISBN: 978-3831600953. (Forschungsberichte IWB, Nr. 159).

HALPERIN ET AL. 2000
Halperin, D.; Latombe, J.-C.; Wilson, R. H.: A General Framework for Assembly Planning: The Motion Space Approach. Algorithmica 26 (2000) 3-4, S. 577-601.

HAMMERSTINGL & REINHART 2015
Hammerstingl, V.; Reinhart, G.: Unified Plug&Produce Architecture for Automatic Integration of Field Devices in Industrial Environments. In: IEEE (Hrsg.): IEEE International Conference on Industrial Technology (ICIT). Sevilla, Spanien, 17.-19. März 2015.

HASSELBRING 2006
Hasselbring, W.: Software-Architektur. Informatik-Spektrum 29 (2006) 1, S. 48-52.

HATWIG 2014
Hatwig, J.: Automatisierte Bahnplanung für Industrieroboter und Scanneroptiken bei der Remote-Laserstrahlbearbeitung. Diss., Technische Universität München. München: Utz 2014. ISBN: 9783831644056. (Forschungsberichte IWB, Nr. 289).

HAUN 2007
Haun, M.: Handbuch Robotik. Programmieren und Einsatz intelligenter Roboter. 1. Aufl. Berlin: Springer 2007. ISBN: 354036918X. (VDI-Buch).

HAVIV
Haviv, A. Q.: MEAN web development. Master real-time web application development using a mean combination of MongoDB, Express, Angular JS, and Node.js. Birmingham: Packt Publishing 2014. ISBN: 9781783983285. (Community experience distilled).

HEDELIND & JACKSON 2007
Hedelind, M.; Jackson, M.: The Need for Reconfigurable Robotic Systems. In: Zäh, M. F. (Hrsg.): 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV 2007). Toronto, Ontario, Canada, 22.-24. Juli 2007. Windsor 2007, S. 1253-1262. ISBN: 0978318706.

HEINRICH ET AL. 2015
Heinrich, B.; Linke, P.; Glöckler, M.: Grundlagen Automatisierung. Sensorik, Regelung, Steuerung. Wiesbaden: Springer Fachmedien 2015. ISBN: 978-3-658-05961-3.

HENSEL 2011
Hensel, T.: Modellbasierter Entwicklungsprozess für Automatisierungslösungen. Diss., Technische Universität München. München: Utz 2011. ISBN: 3831641676. (Forschungsberichte IWB, Nr. 258).

HOFFMANN 2013
Hoffmann, D. W.: Software-Qualität. 2. Aufl. Berlin: Springer Vieweg 2013. ISBN: 3642357008. (eXamen.press).

HOFFMEISTER 2013
Hoffmeister, M.: Modelle zur aufgabengeführten Produktionsausführung in der wandlungsfähigen Produktion. Diss., Universität Stuttgart. Stuttgart: Fraunhofer-Verl. 2013. ISBN: 978-3-8396-0530-1. (Stuttgarter Beiträge zur Produktionsforschung, Nr. 8).

HOLLE 2002
Holle, W.: Rechnerunterstützte Montageplanung. Montageplanung und Simultaneous Engineering. München: Hanser 2002. ISBN: 978-3446219861.

HOLLMANN 2013
Hollmann, R.: Modellbasierte Roboterprogrammierung. In: Verl, A. et al. (Hrsg.): Seminar Mensch-Roboter-Interaktion: Sicherer und intuitiver Umgang mit Industrierobotern. Stuttgart, 28. November 2013.

HU ET AL. 2011
Hu, S. J.; Ko, J.; Weyand, L.; ElMaraghy, H. A.; Lien, T. K.; Koren, Y.; Bley H.; Chryssolouris G.;Nsar, N.; Shpitalni, M.: Assembly system design and operations for product variety. CIRP Annals - Manufacturing Technology 60 (2011) 2, S. 715-733.

HUCKABY & CHRISTENSEN 2012
Huckaby, J.; Christensen, H. I.: A Taxonomic Framework for Task Modeling and Knowledge Transfer in Manufacturing Robotics. In: AAAI Cognitive Robotics Workshop (Hrsg.): Cognitive robotics. Toronto, Canada, 22.-23. Juli. Palo Alto, Calif: AAAI Press 2012. ISBN: 978-157735571-7.

HUMBURGER 1998
Humburger, R.: Konzeption eines Systems zur aufgabenorientierten

Roboterprogrammierung. Als Ms. gedr Aufl. Aachen: Shaker 1998. ISBN: 9783826534270. (Berichte aus der Produktionstechnik, Nr. Bd. 98,4).

HUNDT 2012
Hundt, L.: Durchgängiger Austausch von Daten zur Verhaltensbeschreibung von Automatisierungssystemen. Berlin: Logos-Verlag 2012. ISBN: 3832532587.

IFR 2014
IFR (Hrsg.): World Robotics 2014 - Industrial Robots. <http://www.ifr.org/news/ifr-press-release/accelerating-demand-for-industrial-robots-will-continue-658/> - 07.02.2015.

INTERCAM SA 2015
Intercam SA (Hrsg.): Robotmaster. <http://www.intercamsa.com/whatsnew.php> - 17.02.2015.

ISERMANN 2008
Isermann, R.: Mechatronische Systeme. Grundlagen. 2., vollst. neu bearb. Aufl. Berlin, Heidelberg, New York, NY: Springer 2008. ISBN: 3540325123.

ISO 10303
ISO 10303: Industrielle Automatisierungssysteme und Integration - Produktdatendarstellung und -austausch. Berlin: Beuth Verlag 1994.

ISO 18629
ISO 18629: Industrielle Automatisierungssysteme und Integration - Prozessbeschreibungssprache - Teil 1: Übersicht und grundlegende Festlegungen. Berlin: Beuth Verlag 2004.

ISO/IEC 7498
ISO/IEC 7498: Informationstechnik - Kommunikation Offener Systeme. Berlin: Beuth Verlag 1994.

ISO/PAS 17506
ISO/PAS 17506: Industrielle Automatisierungssysteme und Integration - Festlegung des COLLADA-Schemas für 3D-Visualisierung industrieller Daten. Berlin: Beuth Verlag 2012.

JI ET AL. 2011
Ji, H.; Lenord, O.; Schramm, D.: A Model Driven Approach for Requirements Engineering of Industrial Automation Systems. In: Cellier, F. E. et al. (Hrsg.): Proceedings of the 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools (EOOLT). Zürich, Schweiz, 5. Sept. Linköping: Linköping Univ. Electronic Press 2011, S. 9-18. ISBN: 978-91-7519-825-5.

JOHN & TIEGELKAMP 2009
John, K. H.; Tiegelkamp, M.: SPS-Programmierung mit IEC 61131-3. Konzepte und Programmiersprachen, Anforderungen an Programmiersysteme, Entscheidungshilfen. 4., neubearbeitete Aufl. Dordrecht, New York: Springer 2009. ISBN: 3642002692.

JONAS 2000
Jonas, C.: Konzept einer durchgängigen, rechnergestützten Planung von Montageanlagen. Diss., Technische Universität München: Utz 2000. ISBN: 9783896758705. (Forschungsberichte IWB, Nr. 145).

KAIN ET AL. 2008
Kain, S.; Heuschmann, C.; Schiller, F.: Von der virtuellen Inbetriebnahme zur Betriebsparallelen Simulation. atp edition 50 (2008) 8, S. 48-52.

KASHKOUSH & ELMARAGHY 2014
Kashkoush, M.; ElMaraghy, H.: Consensus tree method for generating master assembly sequence. Production Engineering 8 (2014) 1-2, S. 233-242.

KAUFMAN ET AL. 1996
Kaufman, S. G.; Wilson, R. H.; Jones, R. E.; Calton, T. L.; Ames, A. L.: The Archimedes 2 mechanical assembly planning system. In: IEEE (Hrsg.): IEEE International Conference on Robotics and Automation. Minneapolis, MN, USA, 22.-28. April 1996, S. 3361-3368.

KIEFER 2007
Kiefer, J.: Mechatronikorientierte Planung automatisierter Fertigungszellen im Bereich Karosserierohbau. [Online-Ausg.] Aufl. Saarbrücken: LKT, Univ. 2007. ISBN: 3930429721. (Schriftenreihe Produktionstechnik, Nr. Bd. 43).

KIEFER ET AL. 2009
Kiefer, J.; Ollinger, L.; Bergert, M.: Virtuelle Inbetriebnahme – Standardisierte Verhaltensmodellierung mechatronischer Betriebsmittel im automobilen Karosserierohbau. atp – Automatisierungstechnische Praxis 51 (2009) 7, S. 40-46.

KLEINEIDAM 1990
Kleineidam, G.: CAD, CAP: Rechnergestützte Montagefeinplanung. Diss., Universität Erlangen - Nürnberg. München: Hanser 1990. ISBN: 978-3446161122. (Fertigungstechnik - Erlangen, Nr. 12).

KLUGE 2011
Kluge, S.: Methodik zur fähigkeitsbasierten Planung modularer Montagesysteme.

Diss., Universität Stuttgart: Universitätsbibliothek der Universität Stuttgart 2011. ISBN: 978-3-939890-81-2. (IPA-IAO-Forschung und Praxis, Nr. 510).

KNACKSTEDT ET AL. 2009
Knackstedt, R.; Pöppelbuß, J.; Becker, J.: Vorgehensmodell zur Entwicklung von Reifegradmodellen. (Hrsg.): 9. Internationale Tagung Wirtschaftsinformatik. Wien, Österreich, 25.-27. Februar 2009.

KOCH 2005
Koch, O.: Konzeption eines generischen Vorgehensmodells zur strategieorientierten und partizipativen Einführung komplexer Softwaresysteme unter Berücksichtigung organisatorischer Gestaltungsprozesse. Diss., Universität Kassel: WITEC 2005. ISBN: 978-3981115307.

KOREN ET AL. 1999
Koren, Y.; Heisel, U.; Jovane, F.; Moriwaki, T.; Pritschow, G.; Ulsoy, G.; van Brussel, H.: Reconfigurable Manufacturing Systems. (Hrsg.): CIRP Annals - Manufacturing Technology 1999, S. 527-540.

KRAUSE LIB 2011
Krause LIB GmbH (Hrsg.): Produktübersicht - RobotWorks. <http://www.robotworks-de.com/produkte/produktuebersicht/> - 17.02.2015.

KRÜCKHANS & MEIER 2013
Krückhans, B.; Meier, M.: Industrie 4.0 – Handlungsfelder der Digitalen Fabrik zur Optimierung der Ressourceneffizienz in der Produktion. In: Dangelmaier, W. et al. (Hrsg.): ASIM Fachtagung 2013 Simulation in Produktion und Logistik. Paderborn, 09.-11. Oktober 2013. Paderborn: HNI-Verlagsschriftenreihe 2013 2013, S. 31-40.

KRUG 2013
Krug, S.: Automatische Konfiguration von Robotersystemen (Plug&Produce). Diss., Technische Universität München. München: Utz 2013. ISBN: 978-3-8316-4243-4. (Forschungsberichte IWB, Nr. 270).

KRZYSZTOF 2005
Krzysztof, S.: Automatic Code Generation for PLC Controllers. In: Hutchison, D. et al. (Hrsg.): Computer Safety, Reliability, and Security. Berlin, Heidelberg: Springer 2005, S. 303-316. ISBN: 978-3-540-29200-5.

KUFNER 2012
Kufner, A.: Automatisierte Erstellung von Maschinenmodellen für die Hardware-in-the-Loop-Simulation von Montagemaschinen. Diss., Universität Stuttgart.

Heimsheim: Jost-Jetter 2012. ISBN: 3939890944. (ISW/IPA Forschung und Praxis, Nr. 188).

KUGELMANN 1999
Kugelmann, D.: Aufgabenorientierte Offline-Programmierung von Industrierobotern. Diss., Technische Universität München. München: Utz 1999. ISBN: 9783896756152. (Forschungsberichte IWB, Nr. 127).

KUKA ROBOTER 2015
KUKA Roboter GmbH (Hrsg.): KUKA.Sim. <http://www.kuka-robotics.com/germany/de/products/software/kuka_sim/> - 17.02.2015.

LACOUR 2012
Lacour, F.-F.: Modellbildung für die physikbasierte Virtuelle Inbetriebnahme materialflussintensiver Produktionsanlagen. Diss., Technische Universität München. München: Utz 2012. ISBN: 9783831641628. (Forschungsberichte IWB, Nr. 257).

LAMBRECHT & KRÜGER 2014
Lambrecht, J.; Krüger, J.: Evaluation einer natürlich-räumlichen Benutzerschnittstelle auf Basis von Gesten und Augmented Reality zur Programmierung von Industrierobotern. In: Brecher, C. (Hrsg.): Effiziente Produktion. Als Ms. gedr Aufl. Düsseldorf: VDI-Verl. 2014, S. 11-20. ISBN: 3183689022.

LAU 2010
Lau, C.: Methodik für eine selbstoptimierende Produktionssteuerung. Diss., Technische Universität München. München: Utz 2010. ISBN: 3831640122. (Forschungsberichte IWB, Nr. 238).

LEPERS 2005
Lepers, H.: SPS-Programmierung nach IEC 61131-3. Mit Beispielen für CoDeSys und STEP 7. Poing: Franzis 2005. ISBN: 9783772358012. (PC & Elektronik).

LEVI 1988
Levi, P.: Planen für autonome Montageroboter. Berlin, Heidelberg: Springer 1988. ISBN: 978-3-540-50530-3. (Informatik-Fachberichte, Nr. 191).

LINDEMANN ET AL. 2000
Lindemann, U.; Glander, M.; Grunwald, S.; Reicheneder, J.; Stetter, R.; Zanner, S.: Flexible Integration von Produktentwicklung und Montageplanung. Industrie Management 16 (2000) 1, S. 23-27.

LINDWORSKY 2011
Lindworsky, A.: Teilautomatische Generierung von Simulationsmodellen für den entwicklungsbegleitenden Steuerungstest. Diss., Technische Universität München. München: Utz 2011. ISBN: 3831641250. (Forschungsberichte IWB, Nr. 249).

LJUNGKRANTZ & AKESSON 2007
Ljungkrantz, O.; Akesson, K.: A Study of Industrial Logic Control Programming using Library Components. In: IEEE (Hrsg.): IEEE International Conference on Automation Science and Engineering. Scottsdale, AZ, USA, 22.-25. September 2007, S. 117-122.

LOFERER 2002
Loferer, M.: Rechnergestützte Gestaltung von Montagesystemen. München: Utz 2002. ISBN: 3831601186.

LOSKYLL 2013
Loskyll, M.: Entwicklung einer Methodik zur dynamischen kontextbasierten Orchestrierung semantischer Feldgerätefunktionalitäten. Als Ms. gedr Aufl. Kaiserslautern: Techn. Univ 2013. ISBN: 3943995291. (Fortschritt-Berichte pak, Nr. 25 : Kontextadaptive Automatisierung).

LOTTER & WIENDAHL 2005
Lotter, B.; Wiendahl, H.-P.: Montage in der industriellen Produktion. Optimierte Abläufe, rationelle Automatisierung. Berlin: Springer 2005. ISBN: 978-3540214137.

LOTTER & WIENDAHL 2012
Lotter, B.; Wiendahl, H.-P.: Montage in der industriellen Produktion. Ein Handbuch für die Praxis. 2. Aufl. Berlin: Springer Vieweg 2012. ISBN: 3642290612.

LOTTER 1992
Lotter, B.: Wirtschaftliche Montage. Ein Handbuch für Elektrogerätebau und Feinwerktechnik. 2., erw. Aufl. Düsseldorf: VDI-Verl. 1992. ISBN: 978-3184007096.

LUCAS & TILBURY 2003
Lucas, M.; Tilbury, D.: A study of current logic design practices in the automotive manufacturing industry. International Journal of Human-Computer Studies 59 (2003) 5, S. 725-753.

LÜDEMANN-RAVIT 2005
Lüdemann-Ravit, B.: Ein System zur Automatisierung der Planung und

Programmierung von industriellen Roboterapplikationen. Diss., Universität Dortmund. Düsseldorf: VDI-Verl. 2005. ISBN: 3-18-340020-0.

LUDER ET AL. 2013
Luder, A.; Schmidt, N.; Helgermann, S.: Lossless exchange of graph based structure information of production systems by AutomationML. In: Seatzu, C. (Hrsg.): Proceedings of 2013 IEEE 18th International Conference on Emerging Techologies & Factory Automation. Cagliari, Italy, 10.-13. September 2013, S. 1-4. ISBN: 978-1-4799-0864-6.

LUETH 1992
Lueth, T. C.: Automated planning of robot workcell layouts. In: IEEE (Hrsg.): IEEE International Conference on Robotics and Automation. Nice, France, 12-14 Mai 1992, S. 1103-1108.

MAFFAI ET AL. 2011
Maffai, A.; Akilliguglu Hakan; Neves, P.; Ferreira, J.; Onori Mauro: Emerging Behavior as driver for the sustainability of a modular, “skills-centric” production system. (Hrsg.): AFRICON conference. Livingstone, Sambia, 13.-15. Sept. Piscataway, N.J: IEEE 2011, S. 1-6. ISBN: 978-1-61284-992-8.

MALEC ET AL. 2007
Malec, J.; Nilsson, A.; Nilsson, K.; Nowaczyk, S.: Knowledge-Based Reconfiguration of Automation Systems. (Hrsg.): 2007 IEEE International Conference on Automation Science and Engineering. Scottsdale, AZ, USA, 22.-25. Sept. 2007, S. 170-175.

MALEC ET AL. 2013
Malec, J.; Nilsson, K.; Bruyninckx, H.: Describing Assembly Tasks in Declarative Way. In: IEEE (Hrsg.): 2013 IEEE International Conference on Robotics and Automation (ICRA). Karlsruhe, 6.-10. Mai 2013.

MASCLE 2002
Mascle, C.: Feature-based assembly model for integration in computer-aided assembly. Robotics and Computer-Integrated Manufacturing 18 (2002) 5-6, S. 373-378.

MEYNARD 2000
Meynard, F. P.: Control of industrial robots through high-level task programming. Diss., Linköpings Universitet: Linköpings Universitet 2000. ISBN: 9789172197015. (Linköping studies in science and technology. Theses, Nr. 820).

MICHNIEWICZ & REINHART 2014
Michniewicz, J.; Reinhart, G.: Cyber-physical Robotics – Automated Analysis, Programming and Configuration of Robot Cells based on Cyber-physical-systems. Procedia Technology 15 (2014), S. 567-576.

MIEHE ET AL. 2010
Miehe, A.; Müller, N.: Das DIN-Merkmallexikon – Genormte Merkmale für eine eindeutige Produktbeschreibung. Institutsmitteilung N. 35. IMW. TU Clausthal 2010. <https://www.imw.tu-clausthal.de/fileadmin/Forschung/InstMitt/2010/151_158_2010_Das_DIN-Merkmallexikon_genormte_Merkmale_f%C3%BCr_eine_eindeutige_Produktbeschreibung.pdf> - 09.02.2015.

MORROW & KHOSLA 1995
Morrow, J. D.; Khosla, P. K.: Sensorimotor primitives for robotic assembly skills. In: IEEE (Hrsg.): IEEE International Conference on Robotics and Automation. Nagoya, Japan, 21.-27. Mai 1995, S. 1894-1899.

MORROW & KHOSLA 1997
Morrow, J. D.; Khosla, P. K.: Manipulation task primitives for composing robot skills. (Hrsg.): International Conference on Robotics and Automation. Albuquerque, USA, 20.-25. April 1997, S. 3354-3359.

MOSEMANN & WAHL 2001
Mosemann, H.; Wahl, F. M.: Automatic decomposition of planned assembly sequences into skill primitives. IEEE Transactions on Robotics and Automation 17 (2001) 5, S. 709-718.

MOSEMANN 2000
Mosemann, H.: Beiträge zur Planung, Dekomposition und Ausführung von automatisch generierten Roboteraufgaben. Diss., Universität Braunschweig. Aachen: Shaker 2000. ISBN: 9783826577109. (Fortschritte in der Robotik, Nr. Bd. 6).

MUCKENHIRN 2005
Muckenhirn, R.: Konfigurierbares Leitsystem für modulare Montagezellen am Beispiel von Festplatten. Diss., Universität Stuttgart. Heimsheim: Jost-Jetter 2005. ISBN: 9783936947632. (IPA-IAO-Forschung und Praxis, Nr. Nr. 420).

MÜLLER 2012
Müller, H.: Using S-BPM for PLC Code Generation and Extension of Subject-Oriented Methodology to All Layers of Modern Control Systems. In: van der Aalst,

W. et al. (Hrsg.): S-BPM ONE – Scientific Research. Berlin, Heidelberg: Springer Berlin Heidelberg 2012, S. 182-204. ISBN: 978-3-642-29132-6.

MUNZERT 2010
Munzert, U.: Bahnplanungsalgorithmen für das robotergestützte Remote-Laserstrahlschweißen. Diss., Technische Universität München. München: Utz 2010. ISBN: 978-3-8316-0948-2. (Forschungsberichte IWB, Nr. 234).

NAGAI ET AL. 2007
Nagai, T.; Aramaki, S.; Nagasawa, I.: Representation and Programming for a Robotic Assembly Task Using an Assembly Structure. In: IEEE (Hrsg.): 7th IEEE International Conference on Computer and Information Technology (CIT 2007). Aizu-Wakamatsu, Fukushima, Japan, 16.-19. Oktober 2007, S. 909-914.

NAUMANN ET AL. 2007
Naumann, M.; Wegener, K.; Schraft, R.: Control Architecture for Robot Cells to Enable Plug'n'Produce. In: IEEE (Hrsg.): Proceedings of the 2007 IEEE International Conference on Robotics and Automation. Rom, Italien, 10.-14. April. New York: Ieee Press Books 2007, S. 287-292. ISBN: 1-4244-0601-3.

NAUMANN ET AL. 2010
Naumann, M.; Bengel, M.; Verl, A.: Automatic Generation of Robot Application Using a Knowledge Integration Framework. In: VDE (Hrsg.): 41st International Symposium on Robotics (ISR) and 2010 6th German Conference on Robotics (ROBOTIK). München, 7.-9. Juni 2010, S. 631-638. ISBN: 978-3-8007-3273-9.

NEDBAL 2013
Nedbal, D.: Entwicklung eines Vorgehensmodells für die partizipative Einführung betrieblicher Integrationslösungen. Diss., Johannes Kepler Universität Linz2013.

NETO 2013
Neto, P.: Off-line programming and simulation from CAD drawings: Robot-assisted sheet metal bending. In: IEEE (Hrsg.): 39th Annual Conference of the IEEE Industrial Electronics Society (IECON). Wien, Österreich, 10.-13. November 2013, S. 4235-4240.

NEUMANN & WESTKÄMPER 2014
Neumann, M.; Westkämper, E.: Method for a Situation-based Adaptation and Validation of the Manufacturing Capability of Assembly Systems. In: Procedia CIRP (Hrsg.): Proceedings of the 47th CIRP Conference on Manufacturing Systems. Windsor, Canada, 28.-30. April: Elsevier B.V 2014, S. 118-123.

NII 1986
Nii, H. P.: The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures. The AI Magazine 7 (1986) 2, S. 38-53.

OSTGATHE 2012
Ostgathe, M.: System zur produktbasierten Steuerung von Abläufen in der auftragsbezogenen Fertigung und Montage. Diss., Technische Universität München. München: Utz 2012. ISBN: 9783831642069. (Forschungsberichte IWB, Nr. 265).

PAHL 1990
Pahl, G.: Konstruieren mit 3D-CAD-Systemen. Grundlagen, Arbeitstechnik, Anwendungen. Berlin, Heidelberg: Springer 1990. ISBN: 3642475930.

PAN ET AL. 2010
Pan, Z.; Polden, J.; Larkin, N.; van Duin, S.; Norrish, J.: Recent Progress on Programming Methods for Industrial Robots. In: VDE (Hrsg.): 41st International Symposium on Robotics (ISR) and 2010 6th German Conference on Robotics (ROBOTIK). München, 7.-9. Juni 2010, S. 1-8. ISBN: 978-3-8007-3273-9.

PARK ET AL. 2008
Park, H.-S.; Anh, D. B. HH.; Gyu-Bong L.: Development for automatic control system. In: IEEE (Hrsg.): 2008 Third International Forum on Strategic Technologies (IFOST). Novosibirsk, Russland, 23.-29. Juni 2008, S. 421-424.

PARKER 2013
Parker Electromechanical Automation (Hrsg.): PLCOpen State Machine. <http://www.parkermotion.com/dmxreadyv2/faqsmanager/faqsmanager.asp?question=plcopen-state-machine> - 17.02.2015.

PATRON 2004
Patron, C.: Konzept für den Einsatz von Augmented Reality in der Montageplanung. Diss., Technische Universität München. München: Utz 2004. ISBN: 9783831604746. (Forschungsberichte IWB, Nr. 190).

PAWLEWSKI 2014
Pawlewski, P.: Multimodal Approach to Modeling of Manufacturing Processes. Procedia CIRP 17 (2014), S. 716-720.

PFROMMER ET AL. 2013
Pfrommer, J.; Schleipen, M.; Beyerer, J.: PPRS: Production skills and their relation to product, process, and resource. In: IEEE (Hrsg.): 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA). Cagliari, Italien, 10.-13. September 2013, S. 1-4.

PICHLER ET AL. 2002
Pichler, A.; Vincze, M.; Andersen, H.; Madsen, O.; Hausler, K.: A method for automatic spray painting of unknown parts. In: IEEE (Hrsg.): IEEE International Conference on Robotics and Automation. Washington, DC, USA, 11.-15. Mai 2002, S. 444-449.

PLCOPEN 2015
PLCopen (Hrsg.): Benefits for Users of IEC 61131-3. <http://www.plcopen.org/pages/benefits/benefits_for_users_iec61131/> - 09.02.2015.

PRATT 2001
Pratt, M. J.: Introduction to ISO 10303—the STEP Standard for Product Data Exchange. Journal of Computing and Information Science in Engineering 1 (2001) 1, S. 102-103.

QIAN ET AL. 2010
Qian, K.; Fu, X.; Tao, L.: Software architecture and design illuminated. Sudbury, Mass.: Jones and Bartlett Publishers 2010. ISBN: 076375420X. (Jones and Bartlett illuminated series).

RAATZ ET AL. 2008
Raatz, A.; Schütz, D.; Wahl, F. M. (Hrsg.): Robotic systems for handling and assembly. Braunschweig, Germany, 28 - 29 April. Aachen: Shaker 2008. ISBN: 3832271295. (Fortschritte in der Robotik, Nr. 14).

RAUCHENBERGER 2010
Rauchenberger, J.: Reifegradmodelle als Ordnungsrahmen zur systematischen Prozessverbesserung für mechatronische Entwicklungsprozesse. Diss., RWTH. Aachen: Apprimus-Verl 2010. ISBN: 3863590090. (Fertigungsmesstechnik und Qualitätsmanagement, Nr. Bd. 2010,30).

REFA 1985
REFA (Hrsg.): Methodenlehre der Planung und Steuerung. Teil 1: Grundbegriffe. München: Hanser 1985.

REINHART & ZÄH 2007
Reinhart, G.; Zäh, M. F.: Intelligente und effiziente Automatisierung der Montage. wt Werkstattstechnik 97 (2007) 9, S. 602.

REINHART 2000
Reinhart, G.: Deutschland ist Montageland. wt Werkstattstechnik 9.

REINHART ET AL. 2008
Reinhart, G.; Munzert, U.; Vogl, W.: A programming system for robot-based remote-laser-welding with conventional optics. CIRP Annals - Manufacturing Technology 57 (2008) 1, S. 37-40.

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.): 9th IEEE/IAS International Conference on Industry Applications (INDUSCON 2010). Sao Paulo, 8.-10.11.2010 2010, S. 1-6.

REINISCH 1992
Reinisch, H.: Planungs- und Steuerungswerkzeuge zur impliziten Geräteprogrammierung in Roboterzellen. München: C. Hanser 1992. ISBN: 978-3446173804. (Fertigungstechnik--Erlangen, Nr. 31).

REIS 2015
Reis Robotics (Hrsg.): RobOffice. <http://www.reisrobotics.de/Downloads/roboffice> - 17.02.2015.

REUTER 2012
KUKA AG (Hrsg.): KUKA AG. Automatica 2012. <http://www.kuka-ag.de/res/AG/presentations/2012/presentation_120522_de.pdf> - 23.06.2015.

REUTER 2013
Reuter, A.: Definition eines mechatronischen Informationsmodells zur Modellierung von Automatisierungskomponenten und Maschinen. Diss., Universität Stuttgart: Universitätsbibliothek der Universität Stuttgart 2013. ISBN: 3843911371.

RIBO & BRANDNER 2005
Ribo, M.; Brandner, M.: State of the art on vision-based structured light systems for 3D measurements. In: IEEE (Hrsg.): International Workshop on Robotic Sensors: Robotic and Sensor Environments, 2005. Ottawa, ON, Canada, 30. Sept. - 1. Oktober 2005, S. 2-7.

RICHTER & FLÜCKIGER 2007
Richter, M.; Flückiger, M.: Usability Engineering kompakt. Benutzbare Software gezielt entwickeln. 1. Aufl. München: Elsevier 2007. ISBN: 3827418372. (Kompakt-Reihe).

RICHTER 2013
Richter, C.: Reifegradmodelle für Werkzeuglandschaften zur Unterstützung von ITSM-Prozessen. Dissertation, LMU München 2013.

RIDWAN ET AL. 2012
Ridwan, F.; Xu, X.; Liu, G.: A framework for machining optimisation based on STEP-NC. Journal of Intelligent Manufacturing 23 (2012) 3, S. 423-441.

RIESELER 1992
Rieseler, H.: Roboterkinematik — Grundlagen, Invertierung und Symbolische Berechnung. Diss., Universität Braunschweig. Wiesbaden: Vieweg+Teubner Verlag 1992. ISBN: 978-3-528-06515-7.

ROCKWELL AUTOMATION 2015
Rockwell Automation, I. (Hrsg.): RSLogix PLC Programming - Production Control.
<http://www.rockwellautomation.com/rockwellsoftware/products/rslogix.page> - 24.06.2015.

RÖHRDANZ 1998
Röhrdanz, F.: Modellbasierte automatisierte Greifplanung. Diss., Universität Braunschweig. Aachen: Shaker 1998. ISBN: 9783826534478. (Fortschritte in der Robotik, Nr. 3).

ROKOSSA 1999
Rokossa, D.: Prozessorientierte Offline-Programmierung von Industrierobotern. Diss., Universität Dortmund: Shaker Verlag 1999.

ROKOSSA 2011
Rokossa, D.: Effiziente Roboterprogrammierung unter Verwendung wissensintegrierter Funktionsmodule. (Hrsg.): Kongress, Mess- und Automatisierungstechnik; Automation 2011. Baden-Baden, 28.-29. Juni. Düsseldorf: VDI-Verl. 2011, S. 241-244. ISBN: 3180921439.

ROTHHÖFT 2014
Rothhöft, M.: Marktstudie SPS-Systeme. Ergebnisse einer bundesweiten Befragung von Unternehmen aus den Bereichen Maschinenbau und Ingenieurbüros im Januar/Februar 2014. VDMA2014.

RUDOLF 2007
Rudolf, H.: Wissensbasierte Montageplanung in der digitalen Fabrik am Beispiel der Automobilindustrie. Diss., Technische Universität München. München: Utz 2007. ISBN: 3831606978. (Forschungsberichte IWB, Nr. 204).

RUMPE 2004
Rumpe, B.: Modellierung mit UML. Sprache, Konzepte und Methodik. Berlin: Springer 2004. ISBN: 9783540209041. (Xpert.press).

SATTLER & SAAKE 2010
Sattler, K.-U.; Saake, G.: Algorithmen und Datenstrukturen. Eine Einführung mit Java. 4. Aufl. Heidelberg: Dpunkt 2010. ISBN: 978-3-89864-663-5.

SCHACK 2008
Schack, R.: Methodik zur bewertungsorientierten Skalierung der Digitalen Fabrik. Diss., Technische Universität München. München: Utz 2008. ISBN: 9783831607488. (Forschungsberichte IWB, Nr. 207).

SCHAICH 2001
Schaich, C.: Informationsmodell zur fachübergreifenden Beschreibung intelligenter Produktionsmaschinen. Diss., Technische Universität München. München: Utz 2001. ISBN: 9783831600809. (Informationstechnik im Maschinenwesen, Nr. 15).

SCHÄLING 2013
Schäling, B. (Hrsg.): Der moderne Softwareentwicklungsprozess mit UML. <http://www.highscore.de/uml/> - 08.02.2015.

SCHEDL 2009
Schedl, S.: Integration von Anforderungsmanagement in den mechatronischen Entwicklungsprozess. Diss., Technische Universität München. München: Utz 2009. ISBN: 3831608741. (Forschungsberichte IWB, Nr. 229).

SCHEER 2001
Scheer, A.-W.: ARIS - Modellierungsmethoden, Metamodelle, Anwendungen. 4. Aufl. Berlin [u.a.]: Springer 2001. ISBN: 3540416013.

SCHENCK & WILSON 1994
Schenck, D.; Wilson, P. R.: Information modeling the EXPRESS way. New York: Oxford University Press 1994. ISBN: 9781280442865.

SCHMELZER & SESSELMANN 2008
Schmelzer, H. J.; Sesselmann, W.: Geschäftsprozessmanagement in der Praxis. Kunden zufrieden stellen - Produktivität steigern - Wert erhöhen. 6. Aufl. München: Hanser 2008. ISBN: 3446410023.

SCHMIDT 1992
Schmidt, M.: Konzeption und Einsatzplanung flexibel automatisierter Montagesysteme. Diss., Technische Universität München. Berlin, Heidelberg: Springer 1992. ISBN: 364277217X. (Forschungsberichte IWB, Nr. 41).

SCHNELL 2013
Schnell, G.: Bussysteme in der Automatisierungstechnik: Grundlagen und Systeme der industriellen Kommunikation: Vieweg+Teubner Verlag 2013. ISBN: 9783322942586.

SCHOU ET AL. 2013
Schou, C.; Damgaard, J. S.; Bogh, S.; Madsen, O.: Human-robot interface for instructing industrial tasks using kinesthetic teaching. In: IEEE (Hrsg.): 44th International Symposium on Robotics (ISR). Seoul, Süd Korea, 24.-26. Oktober 2013, S. 1-6.

SCHRAFT & MEYER 2006
Schraft, R. D.; Meyer, C.: The need for an intuitive teaching method for small and medium enterprises. In: VDI-Wissensforum GmbH et al. (Hrsg.): Proceedings of the Joint Conference on Robotics. München, 15.-17. Mai. Düsseldorf: VDI-Verl 2006, S. 95-105. ISBN: 3-18-091956-6.

SCHUMANN ET AL. 2011
Schumann, M.; Schenk, M.; Schmucker, U.; Saake, G.: Digital Engineering - Herausforderungen, Ziele und Lösungsbeispiele. In: Schenk, M. (Hrsg.): Digitales Engineering und virtuelle Techniken zum Planen, Testen und betreiben technischer Systeme. Magdeburg, 28.–30. Juni 2011, S. 199-205.

SCHUSTER 1992
Schuster, G.: Rechnergestütztes Planungssystem für die flexibel automatisierte Montage. Diss., Technische Universität München. Berlin, Heidelberg: Springer 1992. ISBN: 366209696X. (Forschungsberichte IWB, Nr. 55).

SCHWARZ 2009
Schwarz, K.: IEC 61850 und IEC 61499 – Als Basis für die intelligente verteilte Automatisierung in elektrischen Energieversorgungssystemen. In: VDE (Hrsg.): Internationaler ETG-Kongress 2009. Düsseldorf, 27.-28. Oktober: VDE-Verlag 2009.

SECKNER 2008
Seckner, M.: Unterstützung automatisierter Mikroproduktion durch wandlungsfähige Montageanlagen. Konzeption und Realisierung einer flexiblen und reaktiven Mikromontage. Als Ms. gedr Aufl. Diss., Technische Universität Kaiserslautern2008. ISBN: 3939432911. (Fortschritt-Berichte pak, Nr. 19 : Mikromontage).

SELIG 2011
Selig, A.: Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten. Diss., Universität Stuttgart. Heimsheim: Jost-Jetter 2011. ISBN: 978-3939890744.

SHNEIDERMAN & PLAISANT 2009
Shneiderman, B.; Plaisant, C.: Designing the user interface. 5th ed. Aufl. Boston: Addison-Wesley 2009. ISBN: 9780321537355.

SIEMENS AG 2006
Siemens AG (Hrsg.): SIMATIC - Programmieren mit STEP 7. Handbuch. <https://www.automation.siemens.com/doconweb/pdf/SINUMERIK_SINAMICS_04_2010_D/S7P.pdf?p=1>.

SIEMENS AG 2015
Siemens AG (Hrsg.): SIMATIC STEP 7 Software - Software für SIMATIC Controller. <http://w3.siemens.com/mcms/simatic-controller-software/de/step7/seiten/default.aspx> - 17.02.2015.

SIEMENS PLM 2009
Siemens PLM Software (Hrsg.): Process Simulate Robotics. <http://www.plm.automation.siemens.com/de_de/products/tecnomatix/robotics_automation/process_simulate.shtml> - 17.02.2015.

SIEMENS PLM 2014
Siemens PLM Software (Hrsg.): Mechatronics Concept Designer. <http://www.plm.automation.siemens.com/de_de/products/nx/for-design/mechatronics-design/index.shtml> - 09.02.2015.

SIMON 2002
Simon, R.: Gerätemodell zur Feldinstrumentierung von Verteilten Automatisierungssystemen (Device Model for Field Instrumentation in Distributed Control Systems). at - Automatisierungstechnik 50 (2002) 10, S. 490-499.

SMALE & RATCHEV 2010
Smale, D.; Ratchev, S.: Application of a Reconfiguration Methodology for Multiple Assembly System Reconfigurations. In: Ratchev, S. (Hrsg.): Precision Assembly Technologies and Systems. Chamonix, Frankreich, 14.-17. Februar. Berlin, Heidelberg: Springer 2010, S. 239-246. ISBN: 978-3-642-11597-4.

SMALE 2011
Smale, D.: Towards an integrated framework for the configuration of modular micro assembly systems. PhD thesis, University of Nottingham. 2011.

SMEROBOTICS 2015
SMErobotics (Hrsg.): Bringing cognitive robotics to the shopfloor - SMErobotics. <http://www.smerobotics.org/> - 13.02.2015.

SMEROBOT™ 2015
SMErobot™ (Hrsg.): SMErobot™ - Homepage. <http://www.smerobot.org/deutsch/> - 13.02.2015.

SOLVANG ET AL. 2007
Solvang, B.; Korondi, P.; Sziebig, G.; Ando, N.: SAPIR: Supervised and Adaptive Programming of Industrial Robots. In: IEEE (Hrsg.): International Conference on Intelligent Engineering Systems, 2007. Budapest, Ungarn, 29. Juni - 2. Juli 2007, S. 281-286.

SPUR & KRAUSE 1997
Spur, G.; Krause, F.-L.: Das virtuelle Produkt: Management der CAD-Technik. München: Hanser 1997. ISBN: 9783446191761.

SPUR 2001
Spur, G.: Ganzheitlicher Ansatz zur virtuellen Produktionsplanung. ZWF - Zeitschrift für wirtschaftlichen Fabrikbetrieb 96 (2001) 1/2, S. 11-16.

STAHN & WILMERSTAEDT 1991
Stahn, G.; Wilmerstaedt, T.: Arbeitswissenschaftliche Gestaltungsschwerpunkte bei der Einführung rechnerunterstützter Arbeitsweisen in die Produktionsvorbereitung. Fertigungstechnik und Betrieb 41 (1991) 1, S. 35-38.

STARKE 1990
Starke, P. H.: Analyse von Petri-Netz-Modellen. Wiesbaden: Vieweg+Teubner Verlag 1990. ISBN: 3663092623.

STICH ET AL. 2013
Stich, P.; Hefner, F.; Backhaus, J.; Drescher, B.; Reinhart, G.: Aufgabenorientierte Entwicklung mechatronischer Produktionssysteme. In: Züricher Hochschule für Angewandte Wissenschaften, Institut für Mechatronische Systeme (Hrsg.): Internationales Forum Mechatronik - Intelligente mechatronische Systeme. Winterthur, 30.-31. Oktober 2013. ISBN: 978-3-033-04189-9.

SUGI ET AL. 2003
Sugi, M.; Maeda, Y.; Aiyama, Y.; Harada, T.; Arai, T.: A holonic architecture for easy reconfiguration of robotic assembly systems. IEEE Transactions on Robotics and Automation 19 (2003) 3, S. 457-464.

SUN ET AL. 2013
Sun, Y. L.; Li, Y.; Zhang, J.; Xu, Z. J.: An Automatic Product Assembly Method Based on Assembly Feature Pair. Advanced Materials Research 655-657 (2013), S. 1697-1701.

SZULMAN ET AL. 2005
Szulman, P.; Hefke, M.; Trifu, A.; Soto, M.; Assmann, D.; Doerr, J.; Eisenbarth, M.: Using ontology-based reference models in digital production engineering integration. In: Zítek, P. (Hrsg.): Proceedings of the 16th IFAC World Congress. Prag, Tschechien, 4. Juli: IFAC, Elsevier 2005, S. 1458-1464.

TEKOUO, W. B. 2013
Tekouo Moutchiho, W.: A New Programming Approach for Robot-based Flexible Inspection systems. Diss., Technische Universität München. München: Utz 2013. (Forschungsberichte IWB, Nr. 272).

THIEL ET AL. 2008
Thiel, K.; Meyer, H.; Fuchs, F.: MES - Grundlage der Produktion von morgen. Effektive Wertschöpfung durch die Einführung von Manufacturing Execution Systems. München: Oldenbourg 2008. ISBN: 978-3835631403.

THOMAS 2008
Thomas, U.: Automatisierte Programmierung von Robotern für Montageaufgaben. Diss., Universität Braunschweig. Aachen: Shaker 2008. ISBN: 3832271015. (Fortschritte in der Robotik, Nr. Bd. 13).

THOMAS ET AL. 2013
Thomas, U.; Hirzinger, G.; Rumpe, B.; Schulze, C.; Wortmann, A.: A new skill based robot programming language using UML/P Statecharts. In: IEEE (Hrsg.): 2013 IEEE International Conference on Robotics and Automation (ICRA). Karlsruhe, 6.-10. Mai 2013, S. 461-466.

TÖNSHOFF ET AL. 1992
Tönshoff, H. K.; Menzel, E.; Park, H. S.: A Knowledge-Based System for Automated Assembly Planning. CIRP Annals - Manufacturing Technology 41 (1992) 1, S. 19-24.

VDA 2010
VDA (Hrsg.): 3D Datenaustausch in der Fabrikplanung. Leitfaden. <https://www.vda.de/de/services/Publikationen/Publikation.~732~.html> - 17.02.2015.

VDI-RICHTLINIE 2206
VDI-RICHTLINIE 2206: Entwicklungsmethodik für mechatronische Systeme. Berlin: Beuth Verlag 2004.

VDI-RICHTLINIE 2219
VDI-RICHTLINIE 2219: Informationsverarbeitung in der Produktentwicklung - Einführung und Wirtschaftlichkeit von EDM/PDM-Systemen. Berlin: Beuth Verlag 2002.

VDI-RICHTLINIE 2221
VDI-RICHTLINIE 2221: Methodik zum Entwickeln und Konstruieren technischer Systeme und Produkte. Berling: Beuth Verlag 1993.

VDI-RICHTLINIE 2860
VDI-RICHTLINIE 2860: Montage- und Handhabungstechnik: Handhabungsfunktionen, Handhabungseinrichtungen, Begriffe, Definitionen, Symbole: 1990.

VDI-RICHTLINIE 3633
VDI-RICHTLINIE 3633: Simulation von Logistik-, Materialfluss- und Produktionssystemen. Berlin: Beuth Verlag 2014.

VDI-RICHTLINIE 4499
VDI-RICHTLINIE 4499: Digtale Fabrik. Berlin: Beuth Verlag 2008.

VDI/VDE 2009
VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (Hrsg.): Automation 2020 -Bedeutung und Entwicklung der Automation bis zum Jahr 2020. Thesen und Handlungsfelder.
<http://www.vdi.de/fileadmin/vdi_de/redakteur_dateien/gma_dateien/AT_2020_INTERNET.pdf> - 07.02.2015.

VDI/VDE 3682
VDI/VDE 3682: Formalisierte Prozessbeschreibung. Berlin: Beuth Verlag 2014.

VDI/VDE TECHNISCHE REGEL 3683
VDI/VDE TECHNISCHE REGEL 3683: Beschreibung von Steuerungsaufgaben - Anleitung zum Erstellen eines Pflichtenheftes. Berlin: Beuth Verlag 1986.

VDI/VDE 3695
VDI/VDE 3695: Engineering von Anlagen - Evaluieren und optimieren des Engineerings. Berlin: Beuth Verlag 2010.

VERL & NAUMANN 2008
Verl, A.; Naumann, M.: Plug'n'Produce-Steuerungsarchitektur für Roboterzellen. Automatische Code-Generierung für Roboterzellen aus Prozess- und Geräte-Beschreibungen. wt Werkstattstechnik 98 (2008) (2008) 5, S. S. 384-390.

VISUAL COMPONENTS 2015
Visual Components (Hrsg.): 3DCreate. <http://www.visualcomponents.com/products/3dcreate/> - 17.02.2015.

VOGEL-HEUSER & WANNAGAT 2009
Vogel-Heuser, B.; Wannagat, A.: Modulares Engineering und Wiederverwendung mit CoDeSys V3 für Automatisierungslösungen mit objektorientiertem Ansatz. München: Oldenbourg-Industrieverlag 2009. ISBN: 978-3-8356-3105-2.

VOGL 2009
Vogl, W.: Eine interaktive räumliche Benutzerschnittstelle für die Programmierung von Industrierobotern. Forschungsberichte IWB, Band 228. 1. Aufl. Aufl. Diss., Technische Universität München. München: Utz 2009. ISBN: 9783831608690. (Forschungsberichte IWB, Nr. 228).

WANG ET AL. 2014
Wang, H.; Rong, Y.; Xiang, D.: Mechanical assembly planning using ant colony optimization. Computer-Aided Design 47 (2014), S. 59-71.

WEBER 1996
Weber, C.: What is a feature and What is its Use? – Results of FEMEX Working Group I. (Hrsg.): Proceedings of ISATA 96. Florence, Italien, 3.-6. Juni 1996, S. 109-116.

WEBER 2009
Weber, W.: Industrieroboter. Methoden der Steuerung und Regelung. 2., neu bearb Aufl. München: Hanser 2009. ISBN: 9783446410312.

WECK & BRECHER 2006
Weck, M.; Brecher, C.: Werkzeugmaschinen 4. Automatisierung von Maschinen und Anlagen. 6. Aufl. Berlin, Heidelberg: Springer 2006. ISBN: 978-3-540-45366-6. (VDI-Buch).

WEILKIENS 2008
Weilkiens, T.: Systems engineering mit SysML/UML. Modellierung, Analyse, Design. 2. Aufl. Heidelberg: dpunkt-Verlag 2008. ISBN: 978-3898645775. (Safari Books Online).

WENK 2002
Wenk, M.: Entwicklung eines konfigurierbaren Steuerungssystems für die flexible Sensorführung von Industrierobotern. Bamberg: Meisenbach 2002. ISBN: 9783875251746. (Fertigungstechnik - Erlangen, Nr. 131).

WESTKÄMPER 2006
Westkämper, E.: Research for Adaptive Assembly. In: Westkämper, E. (Hrsg.): First CIRP International Seminar on Assembly Systems. 15.-17. November. Stuttgart: Fraunhofer-IRB-Verl. 2006, S. 9-15. ISBN: 978-3-8167-7213-2.

WESTKÄMPER 2008
Westkämper, E.: Digitales Engineering von Fabriken und Prozessen. In: Gesellschaft für Fertigungstechnik (Hrsg.): Schriftliche Fassung der Vorträge zum Fertigungstechnischen Kolloquium am 10. und 11. September in Stuttgart ; [Tagungsband]. Stuttgart: Gesellschaft für Fertigungstechnik 2008, S. 427-452. ISBN: 3000256237.

WESTKÄMPER ET AL. 2013
Westkämper, E.; Spath, D.; Constantinescu, C.; Lentes, J.: Digitale Produktion. Ergebnisse aus dem Innovationscluster Digitale Produktion. Dordrecht: Springer 2013. ISBN: 3642202594.

WIELAND 1995
Wieland, E.: Anwendungsorientierte Programmierung für die robotergestützte Montage. Diss., Universität Stuttgart. Berlin, Heidelberg: Springer 1995. ISBN: 3662111632. (ISW Forschung und Praxis).

WILDEMANN 2010
Wildemann, H.: Neue Montagekonzepte - Realisierung von Produktordnungssystemen in der Kleinserienmontage komplexer Produkte bei kleinen und mittleren Unternehmen. Lehrstuhl für Betriebswirtschaftslehre – Unternehmensführung, Logistik und Produktion, TU München 2010. <http://www.veu.de/files/abschlussbericht_15973.pdf> - 20.03.2015.

WITSCH & VOGEL-HEUSER 2004
Witsch, D.; Vogel-Heuser, B.: Automatische Codegenerierung aus der UML für die IEC 61131-3. In: Brauer, W. et al. (Hrsg.): Eingebettete Systeme. Berlin, Heidelberg: Springer 2004, S. 9-18. ISBN: 978-3-540-23424-1.

WOHED ET AL. 2006
Wohed, P.; van der Aalst, W. M. P.; Dumas, M.; ter Hofstede, A. H. M.; Russell, N.: On the Suitability of BPMN for Business Process Modelling. In: Hutchison, D.

et al. (Hrsg.): Business Process Management. Berlin, Heidelberg: Springer 2006, S. 161-176. ISBN: 978-3-540-38901-9.

WÖRN 2003
Wörn, H.: Tendenzen in der Fabrikautomation und Robotik. atp – Automatisierungstechnische Praxis 45 (2003) 7, S. 48-53.

WÜNSCH 2008
Wünsch, G.: Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme. Diss., Technische Universität München. München: Utz 2008. ISBN: 9783831607952. (Forschungsberichte IWB, Nr. 215).

YASKAWA EUROPE 2015
YASKAWA Europe GmbH (Hrsg.): MotoSim - MOTOMAN. <http://www.motoman.de/de/produkte/software/product-view/?no_cache=1&tx_catalogbasic_pi1%5Buid%5D=27&cHash=a1a20e86dd502dc18c29a8cacaa2522a> - 23.06.2015.

YE ET AL. 1999
Ye, N.; Banerjee, P.; Banerjee, A.; Dech, F.: A comparative study of assembly planning in traditional and virtual environments. IEEE Transactions on Systems, Man and Cybernetics 29 (1999) 4, S. 546-555.

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 94 (2004) 9, S. 438-441.

ZHA & DU 2002
Zha, X.; Du, H.: A PDES STEP-based model and system for concurrent integrated design and assembly planning. Computer-Aided Design 34 (2002) 14, S. 1087-1110.

ZÖLLER-GREER 2010
Zöller-Greer, P.: Software Architekturen. Grundlagen und Anwendungen. 3. Aufl. Wächtersbach: Composia-Verl. 2010. ISBN: 3981163931. (Wissen & Praxis).

Anhang

A1 Übergreifende Vorgehen zur Programmierung von Industrierobotern und SPS

Die aus der Literatur und den Anwendungsfällen entwickelten Vorgehen zur Programmierung von Industrierobotern (vgl. Abschnitt 4.3) und SPS (vgl. Abschnitt 4.4) sind in diesem Abschnitt zusammengefasst.

Abbildung 57 zeigt das abgeleitete Vorgehen bei der simulationsbasierten Offline-Programmierung von Industrierobotern wohingegen Abbildung 58 das Vorgehen bei der Programmierung einer SPS darstellt.

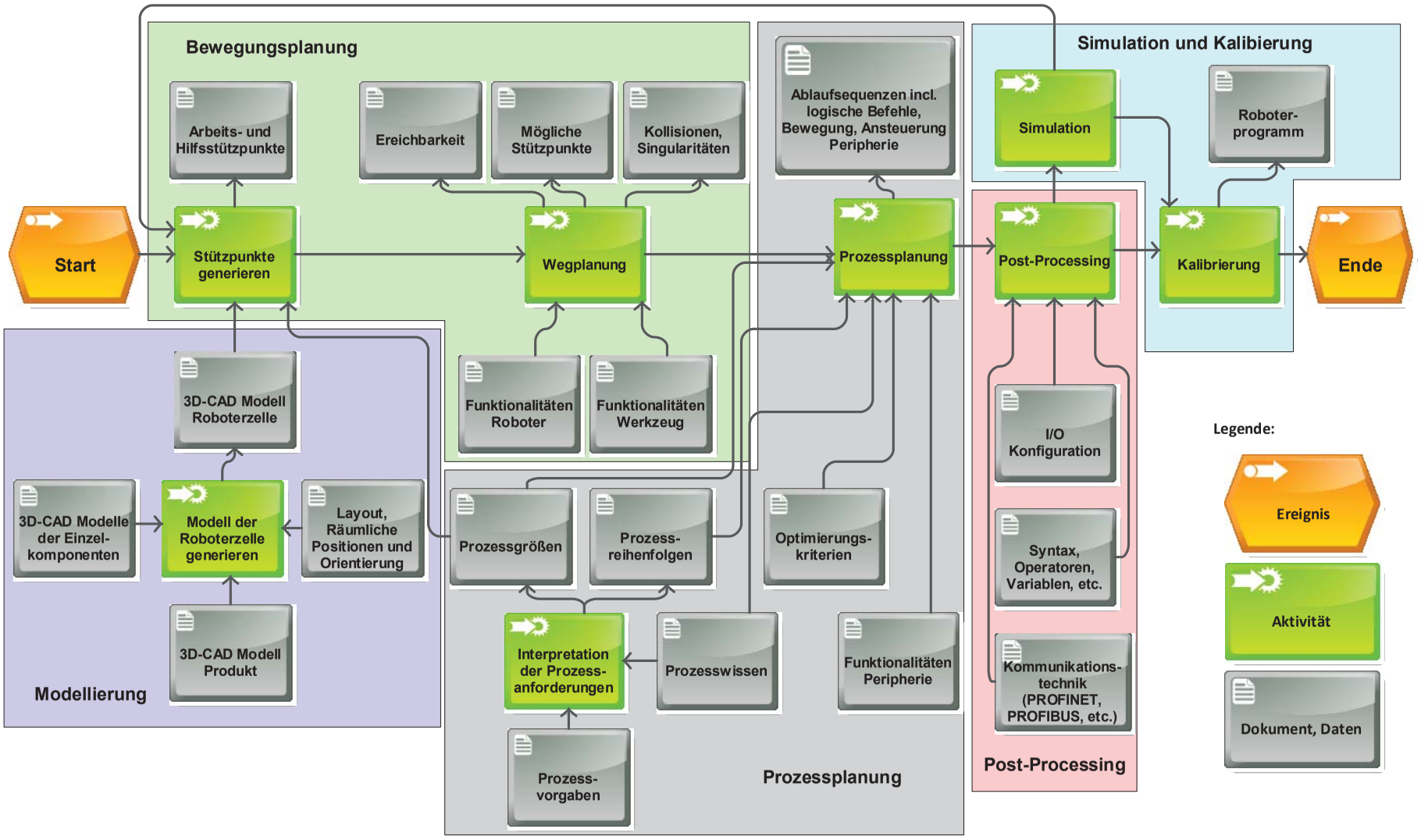


Abbildung 57: Übergreifendes Vorgehen zur simulationsgestützten Offline-Programmierung von Industrierobotern

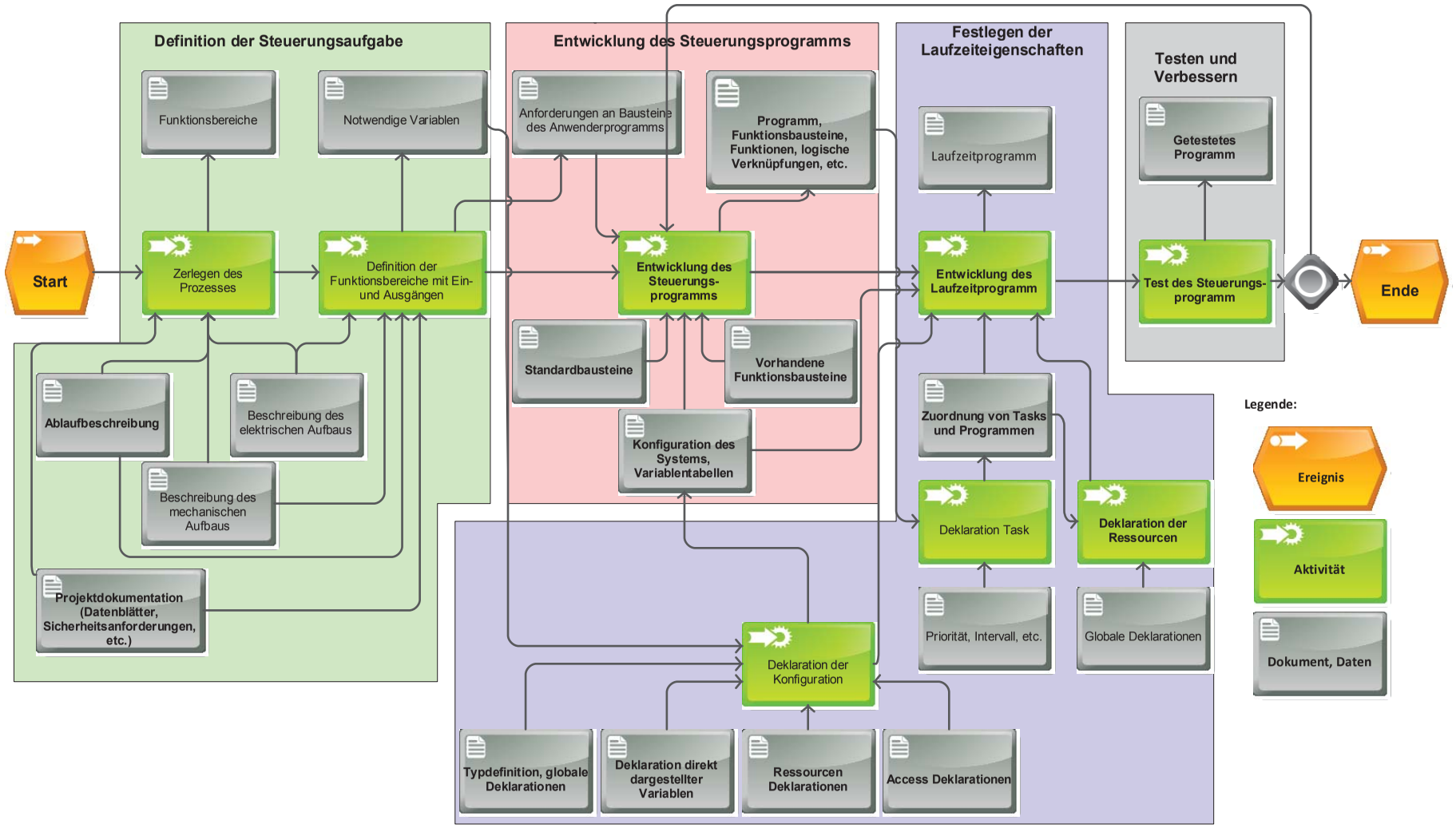


Abbildung 58: Übergreifendes Vorgehen zur Programmierung einer SPS

A2 Klassen von Skills und trennende Merkmale

In diesem Absatz werden die abgeleiteten Klassen von Skills sowie die zugehörigen trennenden Merkmale beschrieben. Es werden jedoch nicht alle Skills inklusive Ein- und Ausgangsgrößen sowie Eigenschaften beschrieben. Deren Darstellung würde die Möglickeiten in dieser Arbeit übersteigen. Die Beschreibung erfolgt jeweils gesondert für die Meta-Klassen „Speichern“, „Bewegen“, „Verbinden“, „Verändern“ und „Vergleichen“.

Speichern:

Tabelle 7: Trennende Merkmale der Meta-Klasse „Speichern“

Trennendes Merkmal	Beschreibung
Hilfsaktion	Damit wird unterschieden, welchen Zweck das Speichern des Objektes hat bzw. ob dieses Grundlage für die Verwendung eines anderen Skills ist. Beispielsweise ist für das Bewegen oder Fügen eines Objektes das Aufrechterhalten der Position und Orientierung durch Speichern notwendig.
Wirkprinzip	Es wird das Wirkprinzip des Skills unterschieden, eingeteilt in die Bereiche Schwerkraft, mechanisch, pneumatisch, magnetisch und adhäsiv.
Freiheitsgrade (DOF)	Unterscheidung nach der Anzahl der durch das Speichern festgelegten translatorischen und rotatorischen Freiheitsgrade des betrachteten Objekts.

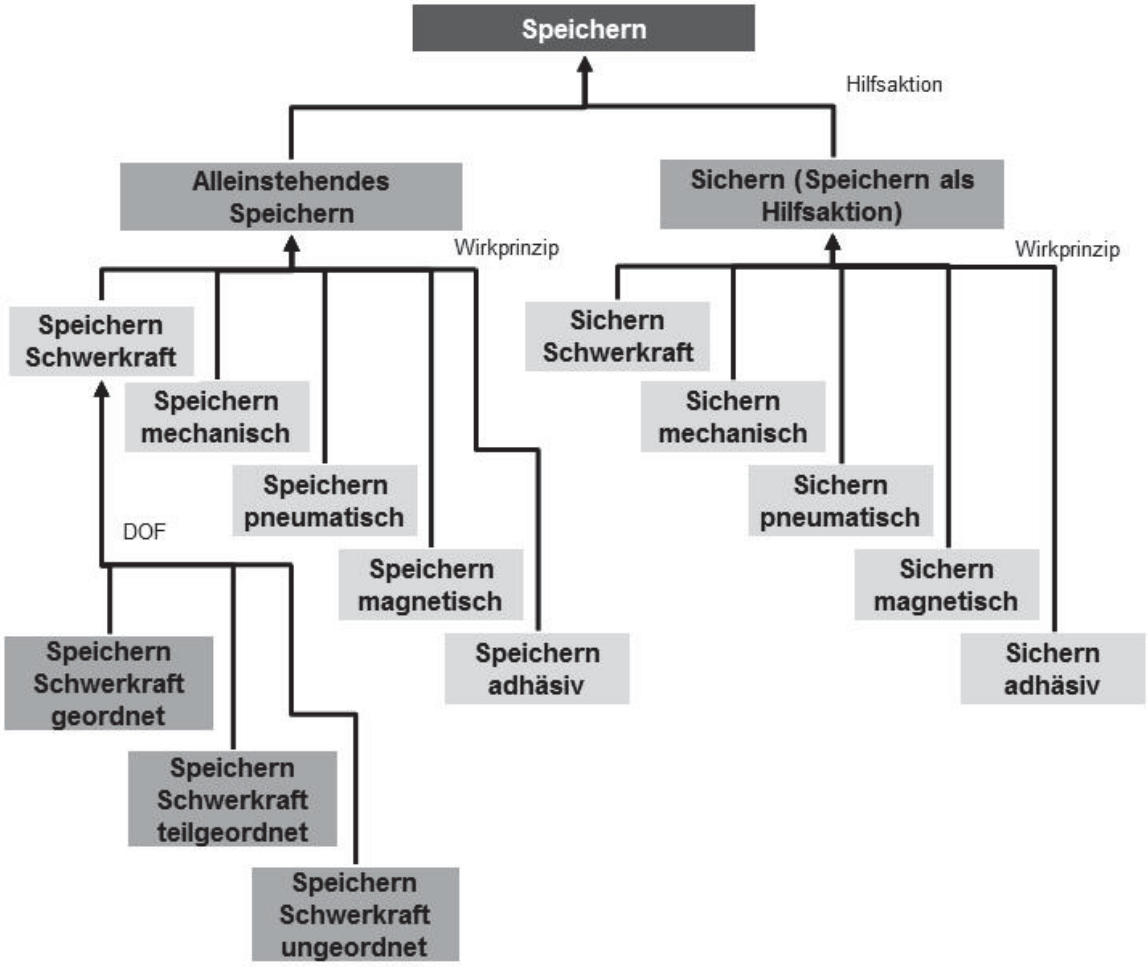


Abbildung 59: Klassendiagramm des Meta-Skills „Speichern“

Bewegen:

Tabelle 8: Trennende Merkmale der Meta-Klasse „Bewegen“

Trennendes Merkmal	**Beschreibung**
Veränderbarkeit der Bahn	Mit diesem Merkmal wird unterschieden, ob die Form der Bahn geändert werden kann oder fest steht.
DOF während der Bewegung	Unterscheidung bzgl. der Anzahl der translatorischen und rotatorischen Freiheitsgrade während der Bewegung, d. h. beim Start- und Endpunkt sowie auf der Bahn der Bewegung.
Geometrische	Mit diesem Merkmal wird die geometrische Form der Bahn

Form der Bewegung	(Gerade, Kreis bzw. beliebige Form durch Spline) unterschieden.
Art des Zustandsübergangs	Mit diesem Merkmal wird die Art des Zustandsübergangs unterschieden.

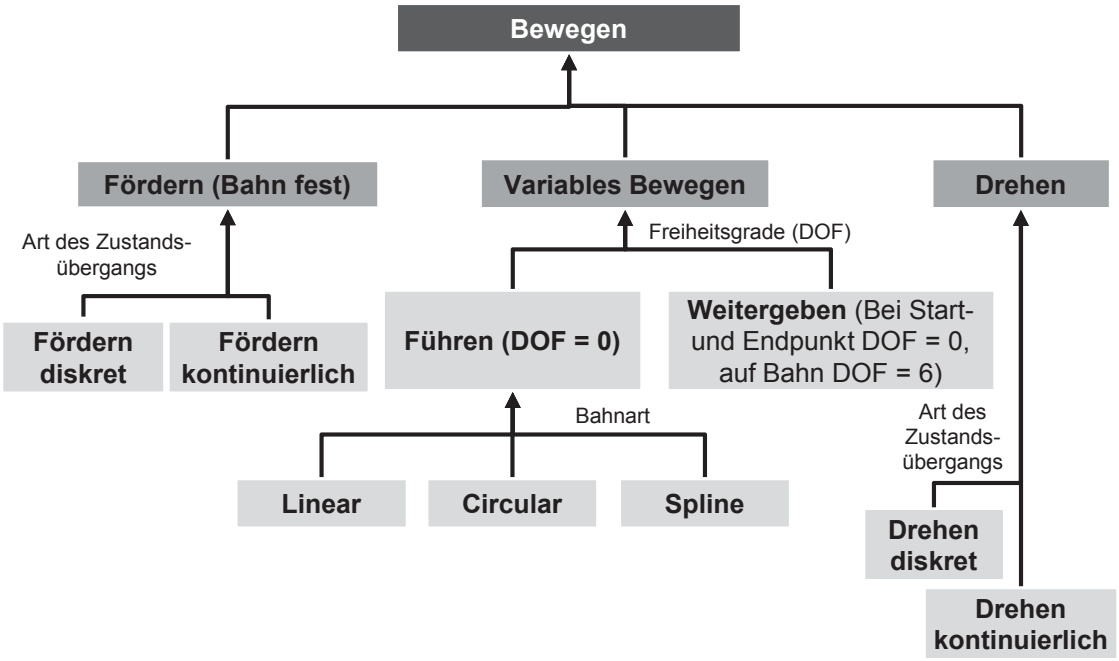


Abbildung 60: Klassendiagramm des Meta-Skills „Bewegen“

Verbinden:

In der DIN 8593 werden die einzelnen Fügeverbindungen nach dem Ordnungsgesichtspunkt (OGP), d. h. nach Art des Zusammenhalts und dessen Erzeugung unterschieden. Die in der DIN 8593 vorgegebene Unterteilung wird weitestgehend übernommen. Die Trennung auf oberster Ebene erfolgt nach der Art des Zusammenhalts. Die trennenden Merkmale unterscheiden sich jedoch teilweise für die Unterklassen.

Tabelle 9: Trennende Merkmale der Meta-Klasse „Verbinden“

Trennendes Merkmal	**Beschreibung**
Art des Zusammenhalts nach DIN 8593	Es werden die Arten Formschluss, Kraftschluss, Stoffschluss (Adhäsion oder Kohäsion) und Einschluss unterschieden.

Wirkmechanismus	Für die Arten des Zusammenhalts wird mit diesem Merkmal der zugehörige Wirkmechanismus unterschieden, welcher den Zusammenhalt erzeugen soll. Beispielsweise gehören Löten und Schweißen zu den stoffschlüssigen Verbindungen. Diese unterscheiden sich jedoch im Wirkmechanismus.
Art der Fügehilfsteile	Mit diesem Merkmale wird betrachtet, welche Art von Fügehilfsteilen für die Verbindung notwendig ist, bzw. ob diese überhaupt verwendet werden. Es werden angelehnt an die DIN 8593-3 folgende Arten unterschieden: Schrauben, Klemmen, Klammern, Nieten, Pass- und Kerbstifte und Nägel
Energieträger nach DIN 8593-6 bzw. 8593-7	Dieses Merkmal unterscheidet die Energieart, die eingesetzt wird, um den Zusammenhalt zu ermöglichen.
Physikalischer Vorgang nach DIN 8593-6	Es wird unterschieden, welcher physikalischer Vorgang es ermöglicht, die Verbindung zu erzeugen.
Art der Verfestigung des Klebstoffes nach DIN 8593-8	Damit werden die Skills dahingehend unterschieden, welche Art der Verfestigung die zugehörigen Klebstoffe besitzen.
Art des Anwendens von Klebeverbindungen nach DIN 8593-8	Mit diesem Merkmal wird unterschieden, welcher Mechanismus für das physikalische Abbinden verwendet wird.
Veränderung der Fügepartner	Es wird unterschieden, ob der jeweilige Skill des Verbindens eine Veränderung der Fügepartner beinhaltet.
Mechanismus des Urformens	Damit wird unterschieden, mit welchem Mechanismus des Urformens die Formschlüssige Verbindung hergestellt wird. Dies erfolgt nach der Einteilung in der DIN 8593-4.

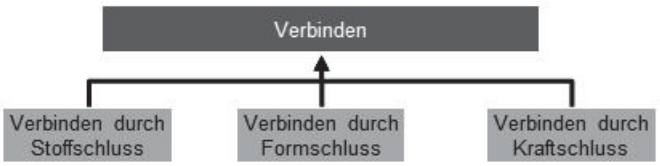


Abbildung 61: Klassendiagramm des Meta-Skills „Verbinden“

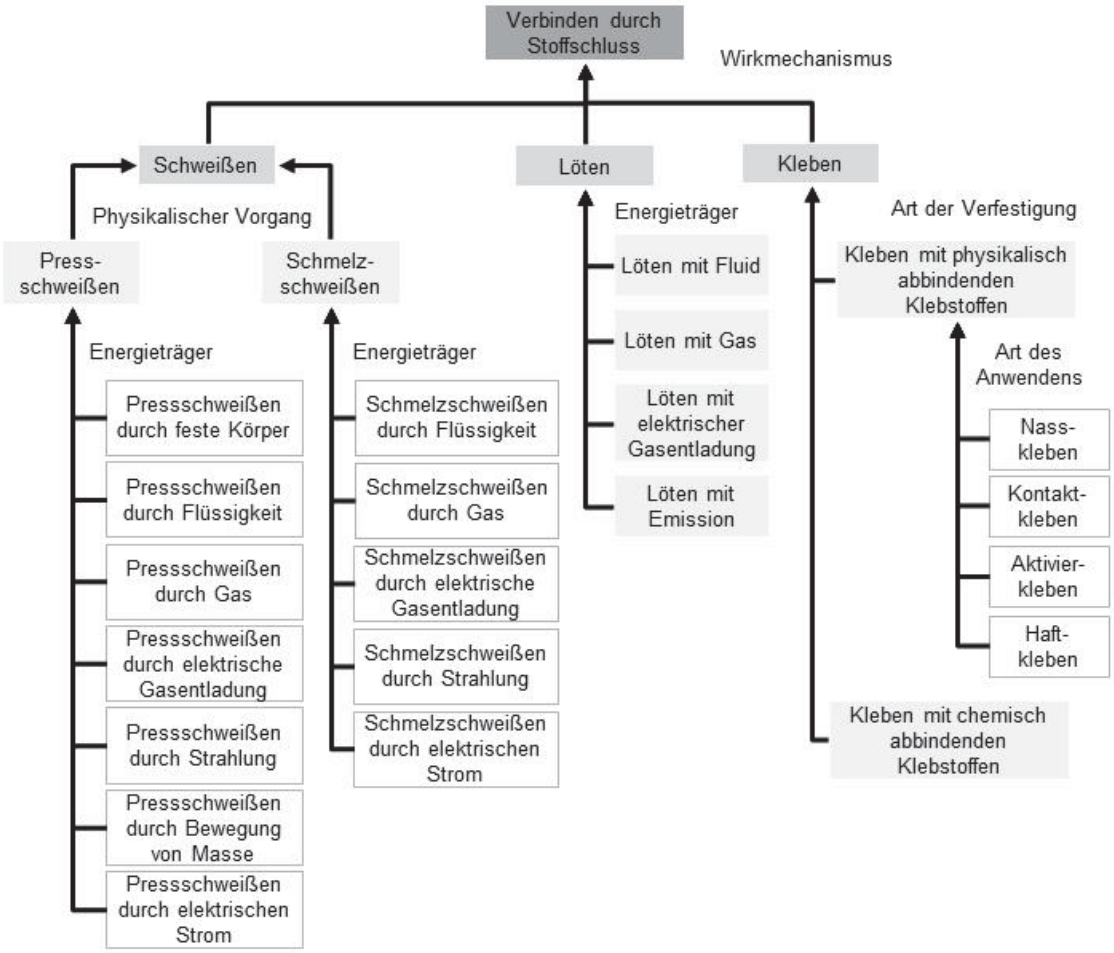


Abbildung 62: Klassendiagramm des Skills „Verbinden durch Stoffschluss“ in Anlehnung an DIN 8593

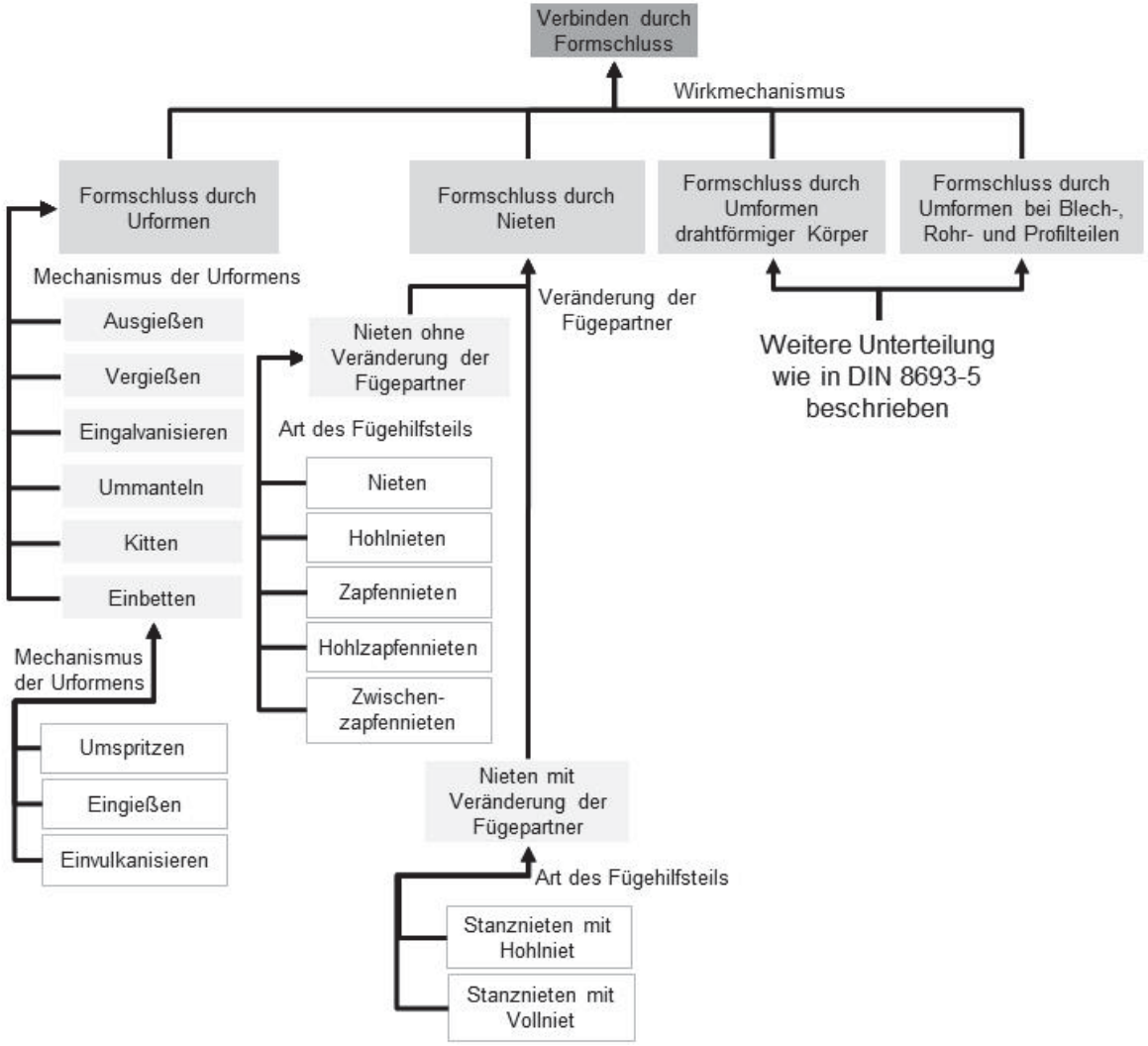


Abbildung 63: Klassendiagramm des Skills „Verbinden durch Formschluss“ in Anlehnung an DIN 8593

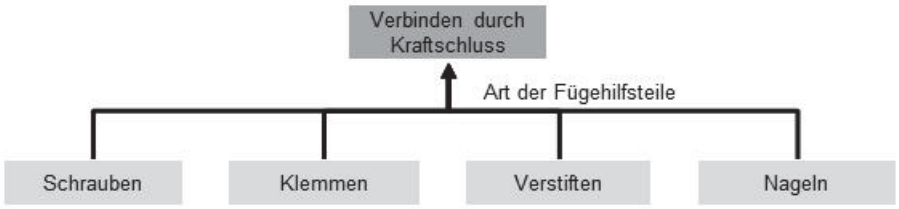


Abbildung 64: Klassendiagramm des Skills „Verbinden durch Kraftschluss“ in Anlehnung an DIN 8593-3

Vergleichen

Tabelle 10: Trennende Merkmale der Meta-Klasse „Vergleichen“

Trennendes Merkmal	Beschreibung
Umfang der Teilbereiche	Damit wird unterschieden, welche Teilbereiche (Erfassen oder/und Vergleichen) der Meta-Klasse des Skills abgedeckt werden.
Erfassungsparameter	Mit diesem Merkmal werden die Klassen nach dem zugehörigem Erfassungsparameter in Anlehnung an SCHMIDT (1992) unterschieden.
Wirkprinzip	Dieses Merkmal trennt die Skills nach den physikalischen und chemischen Wirkprinzipien.

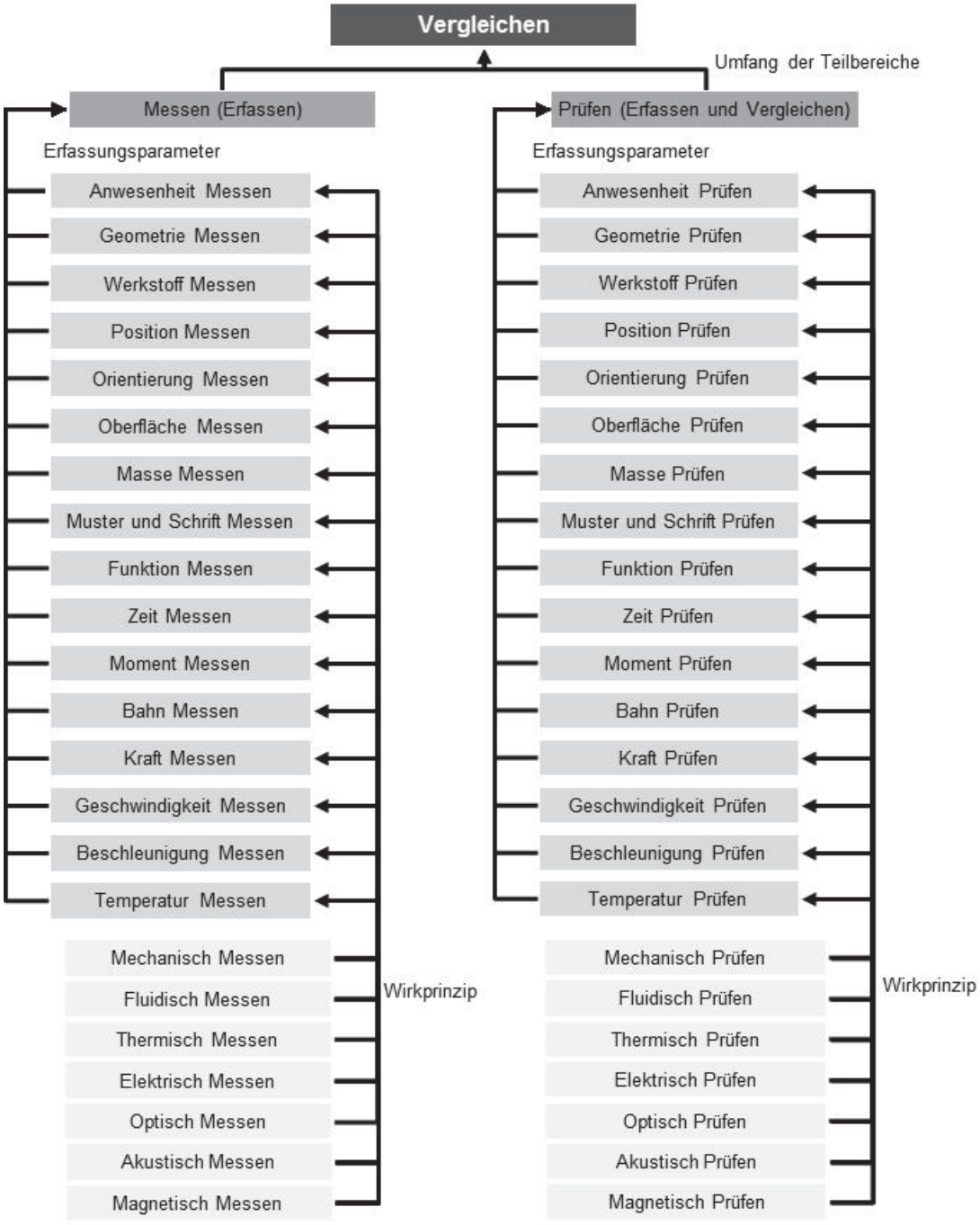


Abbildung 65: Klassendiagramm des Meta-Skills „Vergleichen“ in Anlehnung an SCHMIDT (1992) und VDI 2860

Verändern:

Tabelle 11: Trennende Merkmale der Meta-Klasse „Verändern“

Trennendes Merkmal	Beschreibung
Veränderte Eigenschaft	Mit diesem Merkmal wird die veränderte Eigenschaft in Anlehnung an SCHMIDT (1992) in die Bereiche Formschaffen, Formändern und Stoffändern unterschieden.
Art der Stoffänderung	Die Art der Stoffänderung kann weiter nach deren Art, z. B. dem Erwärmen, unterschieden werden. Darunter fallen auch Skills wie das Bedrucken oder das Ölen.
Art der Formschaffung	Die Art der Formschaffung kann weiter nach deren Art, z. B. dem Giesen, unterschieden.
Art der Formänderung	Die Art der Formänderung unterscheidet nach SCHMIDT (1992) zunächst urformende und trennende Formänderungen.
Art der trennenden Formänderung	Dieses Merkmal unterscheidet die Art der trennenden Formänderung.
Art der urformenden Formänderung	Dieses Merkmal unterscheidet die Art der urformenden Formänderung.

Da für den Meta-Skill „Verändern“ eine sehr umfangreiche Menge an möglichen Unterklassen angegeben werden kann und diese weniger relevant für die Montage sind, erfolgt keine Darstellung der Spezialisierung in einem Klassendiagramm. Beispiele können bei SCHMIDT (1992, S. 37) nachgelesen werden.

A3 Reifegradmodell für die aufgabenorientierte Programmierung

In diesem Teil des Anhangs wird das Reifgradmodell, bestehend aus den Fragen (vgl. Tabelle 12) sowie den Berechnungsvorschriften, beschrieben. Es wird davon ausgegangen, dass alle Unternehmen mindestens Reifegrad eins haben.

Tabelle 12: Fragen des Reifegradmodells

Frage	**Reifegrad**
Themenschwerpunkt Datenmanagement	
Liegen die 3D-Geometriedaten des Produktes digital und maschinenlesbar vor?	2
Liegen weitere produktbeschreibende Daten wie Toleranzen, Oberflächengüte, Masse oder Massenträgheitsmoment digital und maschinenlesbar vor?	3
Ist ein zentrales Datenmanagement der Produktmodelle in Form eines PDM oder PLM-Systems vorhanden?	3
Liegen die Daten bzgl. des geplanten Montageprozesses digital und maschinenlesbar vor?	4
Werden wesentliche Daten des geplanten Montageprozesses in Office-Anwendungen erstellt und verwaltet?	2
Liegt die Fügefolge des Montageprozesses digital und maschinenlesbar vor?	3
Liegen die 3D-Geometriedaten des Montagesystems digital und maschinenlesbar vor?	2
Liegen weiterführende Informationen, wie z. B. Kinematikbeschreibungen, bzgl. der Ressourcen im Montagesystem digital und maschinenlesbar vor?	3
Wesentliche Daten des geplanten Montagesystems werden in Office-Anwendungen erstellt und verwaltet?	2

Sind die digitalen Modelle des Montagesystems in ein PDM/PLM-System integriert?	4
Existiert ein durchgängiges und maschinenlesbares Datenmodell sowohl für Produkt-, Prozess- als auch Montagesysteminformationen?	4
Das Datenmodell für Produkt-, Prozess- als auch Montagesysteminformationen baut auf einem etablierten Standard auf.	4
„Werden für alle Mechanik- und Elektrik-/Elektronik-Komponenten Stucklisten erstellt und verwaltet?“ (RAUCHENBERGER 2010, S. 150)	2
Themenschwerpunkt Entwicklungsprozess	
Gibt es ein nahezu gleichbleibendes Spektrum an angewandten Montageprozessen?	3
Bei den entwickelten Montagesystemen handelt es sich um Serienprodukte mit kleinen Anpassungen?	3
Wird bei der Entwicklung des Montagesystems auf einen Baukastensystem zurückgegriffen?	3
„Werden die Montage- und Bauunterlagen vollständig erstellt?“ (RAUCHENBERGER 2010, S. 150)	2
Die Abstimmung zwischen der Produktentwicklung und der Montageplanung erfolgt über ein Informationssystem?	4
Erfolgt die Entwicklung des Produktes durchgehend rechnergestützt?	2
Erfolgt die Entwicklung des Produktes durchgehend rechnergestützt, inklusive dem Einsatz von Simulationsanwendungen?	3
Einzelne Montageprozesse werden durch Simulationen abgesichert?	3

Die Entwicklung des Montagesystems erfolgt durchgehend rechnergestützt inklusive dem Einsatz von Simulationsanwendungen?	4
Die Absicherung der Inbetriebnahme des Montagesystems erfolgt virtuell mit einem Simulationsmodell?	4
Themenschwerpunkt Mitarbeiterqualifikation	
Besitzen mögliche Anwender des Programmiersystems Grundlagenkenntnisse im Umgang mit Simulations- oder 3D-Konstruktionstools?	3
Besitzen mögliche Anwender des Programmiersystems umfangreiche im Umgang Simulations- und 3D-Konstruktionstools?	4
Besitzt die Mehrheit der späteren Anwender Kenntnisse bzgl. den typischerweise eingesetzten Prozessen?	3
Besitzt die Mehrheit der späteren Anwender ausreichendes Wissen bzgl. dem Aufbau und der Funktionsweise von Montagesystemen?	2
Besitzt die Mehrheit der späteren Anwender Grundlagekenntnisse bzgl. der Programmiersprachen der in den entwickelten Montagesystemen vorkommenden Programmiersprachen?	3
Besitzt die Mehrheit der späteren Anwender ausreichendes Wissen bzgl. dem Aufbau und der Funktionsweise von Montagesystemen?	2
Sind Mitarbeiter offen gegenüber Veränderungen?	3

Für jede Frage in jedem Themenschwerpunkt gibt es vier unterschiedliche Antwortmöglichkeiten, welche in Tabelle 13 dargestellt sind.

Tabelle 13: : Die Antwortmöglichkeiten der Fragen mit den entsprechenden Prozentsätzen inklusive der zugehörenden Bewertungssätze B_i (RAUCHENBERGER 2010)

Antwortmöglichkeit	Entsprechender Prozentsatz	Bewertungssatz B_i
trifft stets zu	> 90% der betrachteten Fälle	100
trifft überwiegend	60–90% der betrachteten Fälle	66
trifft teilweise zu	10–60% der betrachteten Fälle	33
trifft nicht	< 10% der betrachteten Fälle	0

Um die wesentlichen Merkmale weiter herauszustellen, sind die Themenschwerpunkte zueinander gewichtet. Der Themenschwerpunkt „Datenmanagement“ ist im Verhältnis 3:1:1 zu den anderen gewichtet. Der zugehörige Gewichtungsfaktor wird mit G_T bezeichnet. Übergreifend erfolgen die Gewichtung und deren Berechnung in Anlehnung an RAUCHENBERGER (2010, S. 67). Neben den Themenschwerpunkten sind auch den Fragen Gewichtungsfaktoren zugeordnet. „Diese Faktoren nehmen dabei einen Wert zwischen 0 (keine Bedeutung) und 3 (maximale Bedeutung) an. Für die eigentliche Berechnung wird der Gewichtungsfaktor durch den Divisor 3 auf ein Intervall zwischen 0 und 1 normiert.“ RAUCHENBERGER (2010, S. 67). Außerdem werden die so normierten Gewichtungsfaktoren durch die Summe aller Fragen eines Reifegrads in einem Themengebiet geteilt. Dadurch ergeben sich absolut normierte Gewichtungs-faktoren (G_{abs}), welche in Summe eins ergeben RAUCHENBERGER (2010, S. 67).

Die Mittelwerte der Erfüllungsgrade eines Reifegrads in einem Themengebiet berechnen sich damit in Anlehnung an RAUCHENBERGER (2010) zu:

$$E_{ges}^{R} = \sum_{i}^{n_F} E_i^F \times G_{abs_i}^F \qquad \text{(Gl. 1-1)}$$

mit

E_{ges}^{R}: Gesamter Erfüllungsrad eines Reifegrads eines Themengebiets
E_i^F: Erfüllungsrad einer Frage eines zugehörigen Reifegrads
$G_{abs_i}^F$: Absoluter Gewichtungsfaktor einer Frage für einen Reifegrad
n_F: Anzahl der Fragen zu einem Reifegrad in einem Themengebiet

A4 Zuordnung von Benutzergruppen, Informationsarten und technischen Konzepten für Benutzerschnittstellen

Im Rahmen dieser Arbeit wurde sowohl eine Zuordnung von Informationsarten zu technischen Konzepten für die Programmierung bzw. Benutzerschnittstelle als auch der Nutzergruppen zu den technischen Konzepten erarbeitet. Die Zuordnungen sind in den Tabelle 14 und Tabelle 15 dargestellt.

Tabelle 14: Zuordnung von Informationsarten zu technischen Konzepten

technisches Konzept / Informationsart	textuell	visuell	graphisch- bzw. simulations-basiert	Multi-Modal
Geometrie	○	◔	●	●
Ablauf	○	●	◕	●
Parameter	●	●	◕	◔
Vordefinierte Bausteine	○	●	◕	◔
Verwaltung	○	●	◕	◔

○ nicht geeignet ◔ weniger geeignet ◑ durchschnittlich geeignet ◕ geeignet ● vollständig geeignet

Tabelle 15: Eignung von technischen Konzepten von Bedienerschnittstellen zu Nutzergruppen

technisches Konzept / Nutzergruppe	textuell	visuell	graphisch- bzw. simulations-basiert	Multi-Modal
Bediener	◔	◕	◑	●
Anwender	◔	◑	◕	◕
Prozessexperte	◔	◕	●	◔

○ nicht geeignet ◔ weniger geeignet ◑ durchschnittlich geeignet ◕ geeignet ● vollständig geeignet

A5 Blackboard in AutomationML

Abbildung 66 zeigt einen Auszug des Blackboards in AutomationML. Im Speziellen wird der Pointer auf die Funktionsbeschreibung (FunctionMapping) dargestellt. Mit den Attributen des Pointers erfolgt die Zuordnung auf die genauen Bestandteile der extern hinterlegten Funktionsbeschreibung.

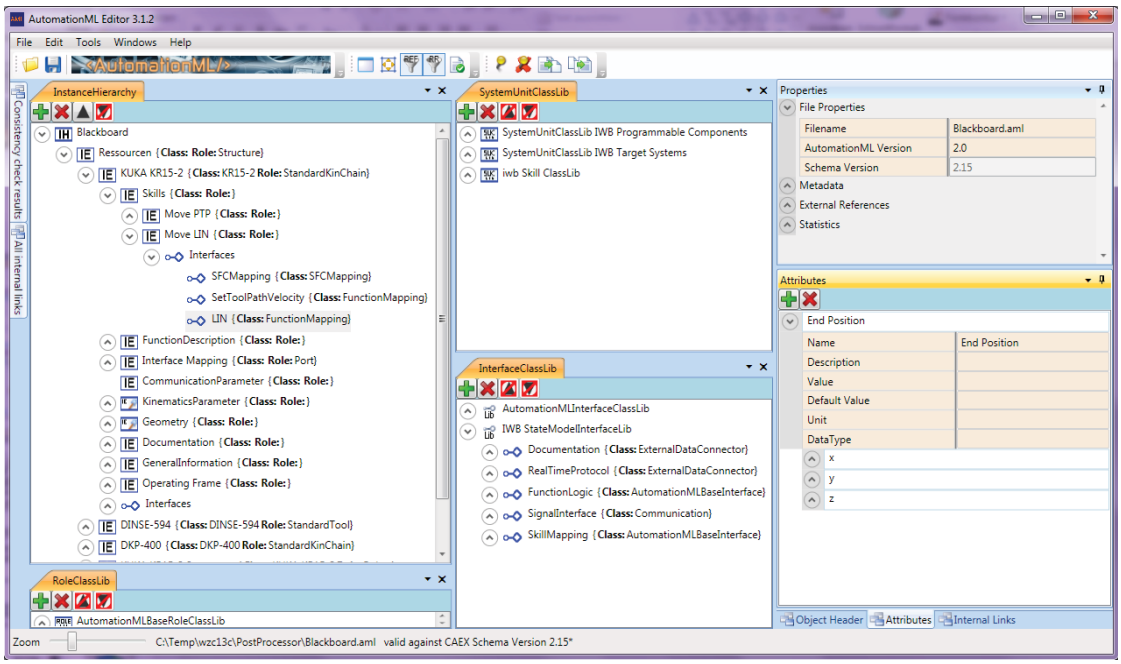


Abbildung 66: Blackboard und Mapping zwischen Skill und Funktionsbeschreibung

Verzeichnis betreuter Studienarbeiten

Der Autor der vorliegenden Dissertation hat in den Jahren von 2011 bis 2015 unter wesentlicher wissenschaftlicher, fachlicher und inhaltlicher Anleitung die im Folgenden aufgeführten studentischen Arbeiten am Institut für Werkzeugmaschinen und Betriebswissenschaften (iwb) der Technischen Universität München (TUM) betreut. Die entstandenen Ergebnisse sind teilweise in das vorliegende Dokument eingeflossen. Der Autor dankt allen Studenten für ihre Unterstützung bei dieser wissenschaftlichen Arbeit und wünscht ihnen alles Gute in ihrer beruflichen Laufbahn als Ingenieur.

Büttner, Stefanie: Konzeption und Implementierung eines Umweltmodells für die aufgabenorientierte Programmierung von Robotersystemen. Semesterarbeit. Bearbeitungszeitraum: 20.08.2012 - 18.02.2013

Ebert, Sven: Konzeption und Implementierung eines modularen Aufgabenplanungssystems für die aufgabenorientierte Programmierung von Montagesystemen. Diplomarbeit. Bearbeitungszeitraum: 01.11.2012 - 30.04.2013

Enzinger, Martin: Konzeption und Implementierung eines aufgabenorientierten Programmiersystems mit Fokus auf das Umweltmodell. Diplomarbeit. Bearbeitungszeitraum: 08.11.2013 - 08.05.2014

Feierle, Alexander: Konzeption und Implementierung der Funktionalitäten zur automatischen Greifplanung im Rahmen der aufgabenorientierten Programmierung von Montagesystemen. Bachelorarbeit. Bearbeitungszeitraum: 29.04.2014 - 28.10.2014

Greger, Sabrina: Strukturierung der Eingabesysteme für die aufgabenorientierte Programmierung von Montagesystemen. Bachelorarbeit. Bearbeitungszeitraum: 07.05.2013 - 06.11.2013

Heuss, Lisa: Konzeption und Implementierung eines erweiterten Planungsmoduls für die aufgabenorientierte Programmierung von Montagesystemen. Bachelorarbeit. Bearbeitungszeitraum: 29.04.2014 - 28.10.2014

Hoffmann, Philipp: Integration, Test und Validierung eines aufgabenorientierten Programmiersystems. Bachelorarbeit. Bearbeitungszeitraum: 07.11.2014 – 27.02.2015

Hüdig, Matthias: Konzeption und Implementierung eines Moduls zur automatischen Planung von Schweißprozessen für die aufgabenorientierte Programmierung von Montagesystemen. Diplomarbeit. Bearbeitungszeitraum: 29.04.2014 - 28.10.2014

Klingshirn, Maximilian: Analyse der Einsatzmöglichkeiten von Roboter-Offline-Simulationssystemen zur aufgabenorientierten Programmierung. Bachelorarbeit. Bearbeitungszeitraum: 29.04.14 - 29.08.2014

Klinker, Kai: Konzeption und Implementierung einer Nutzerschnittstelle für die aufgabenorientierte Programmierung von Montagesystemen. Interdisziplinäres Projekt. Bearbeitungszeitraum: 30.10.2014 – 30.03.2015

Papyan, Diana: Erweiterung eines Planungsmoduls für die automatische Greifplanung. Interdisziplinäres Projekt. Bearbeitungszeitraum: 30.10.2014 – 30.03.2015

Reichle, Johannes: Analyse der Möglichkeiten zur Verarbeitung von Sensordaten in der aufgabenorientierten Programmierung von Montagesystemen und Konzeption der notwendigen Planungsfunktionalität: Bachelorarbeit. Bearbeitungszeitraum: 29.04.2014 - 28.10.2014

Röhler, Marcus: Konzeption und Implementierung eines generischen Aufgabenplanungssystems für die aufgabenorientierte Programmierung von Montagesystemen. Bachelorarbeit. Bearbeitungszeitraum: 28.08.2013 - 27.02.2014

Schultz, Thomas: Erweiterung und Implementierung eines Postprozessors. Interdisziplinäres Projekt. Bearbeitungszeitraum: 30.10.2014 – 30.03.2015

Im Rahmen von Forschungspraktikas der Munich School of Engineering wurden folgende Studenten betreut, welche sich mit Teilaspekten des Programmiersystems beschäftigt haben:

Reiser, Benedikt; Brunn, Fabian; Kibler, Simon; Lechner, Daniel; Wüstner, Christoph; Drost, Felix