Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten

Von der Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik
der Universität Stuttgart
zur Erlangung der Würde eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Abhandlung

vorgelegt von
Dipl.-Ing. Andreas Selig
geboren in Rinteln/Weser

Hauptberichter: Prof. Dr.-Ing. Peter Klemm
Mitberichter: Prof. Dr.-Ing. Dr. h. c. mult. Paul J. Kühn
Tag der Einreichung: 19. Oktober 2009
Tag der mündlichen Prüfung: 23. September 2010

Institut für Steuerungstechnik der Werkzeugmaschinen
und Fertigungseinrichtungen
der Universität Stuttgart
2011

ISW/IPA Forschung und Praxis

Berichte aus dem Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen
der Universität Stuttgart und dem Fraunhofer-Institut für
Produktionstechnik und Automatisierung IPA

Herausgeber: Prof. Dr.-Ing. A. Verl

Institut für Steuerungstechnik
der Werkzeugmaschinen und Fertigungseinrichtungen

Fraunhofer
IPA

Andreas Selig

Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten

Nr. 180

JOST-JETTER VERLAG
Fachverlag · 71296 Heimsheim

D 93

ISBN 978-3-939890-74-4
Jost Jetter Verlag, Heimsheim

Dieses Werk ist urheberrechtlich geschützt. Die dadurch begründeten Rechte, insbesondere die der Übersetzung, des Nachdrucks, des Vortrags, der Entnahme von Abbildungen und Tabellen, der Funksendung, der Mikroverfilmung oder der Vervielfältigung auf anderen Wegen und der Speicherung in Datenverarbeitungsanlagen, bleiben, auch bei nur auszugsweiser Verwertung, vorbehalten. Eine Vervielfältigung dieses Werkes oder von Teilen dieses Werkes ist auch im Einzelfall nur in den Grenzen der gesetzlichen Bestimmungen des Urheberrechtsgesetzes der Bundesrepublik Deutschland vom 9. September 1965 in der jeweils gültigen Fassung zulässig. Sie ist grundsätzlich vergütungspflichtig. Zuwiderhandlungen unterliegen den Strafbestimmungen des Urheberrechtsgesetzes.
© Jost Jetter Verlag, Heimsheim 2011.

Printed in Germany.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Sollte in diesem Werk direkt oder indirekt auf Gesetze, Vorschriften oder Richtlinien (z. B. DIN, VDI, VDE) Bezug genommen oder aus ihnen zitiert worden sein, so kann der Verlag keine Gewähr für die Richtigkeit, Vollständigkeit oder Aktualität übernehmen. Es empfiehlt sich, gegebenenfalls für die eigenen Arbeiten die vollständigen Vorschriften oder Richtlinien in der jeweils gültigen Fassung hinzuzuziehen.

Druck: printsystem GmbH, Heimsheim

Geleitwort des Herausgebers

In der Reihe „ISW/IPA Forschung und Praxis“ wird fortlaufend über Forschungsergebnisse des Instituts für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW) und des Fraunhofer Instituts für Produktionstechnik und Automatisierung (IPA) berichtet. Die Institute stehen unter gemeinsamer Leitung von Professor Alexander Verl und beschäftigen sich sowohl im Bereich der Grundlagenforschung als auch im Bereich der angewandten Forschung mit der Weiterentwicklung und Optimierung der Automatisierung in der Produktionstechnik. Fraunhofer IPA und ISW repräsentieren Deutschlands größte Forschungseinrichtung im Bereich der Produktionsautomatisierung.

Besonderes Augenmerk wird auf die Systeme Werkzeugmaschine und Roboter gelegt. Die Arbeiten des ISW konzentrieren sich im Besonderen auf die Bereiche Numerische Steuerungstechnik, Bewegungssteuerung, Planungs- und Leitsysteme, Softwaretechnik, Simulationstechnik und Antriebstechnik. Am Fraunhofer IPA konzentriert man sich im Bereich der Automatisierungstechnik auf die Themen: Servicerobotik, Industrierobotik, Montageautomatisierung, Orthopädiesysteme, Prozessautomatisierung (insbesondere im Lifesciencebereich), Messtechnik, Prüftechnik, Bildverarbeitung sowie die Automatisierung in der Reinst- und Mikroproduktion. Dabei stehen Grundlagenforschung und anwenderorientierte Entwicklung in einem stetigen Austausch, wodurch ein ständiger Technologietransfer zur Praxis sichergestellt wird.

Die Buchreihe erscheint in zwangloser Folge und stützt sich auf Berichte über abgeschlossene Forschungsarbeiten und Dissertationen. Sie soll dem Ingenieur bei der Weiterbildung dienen und ihm Hilfestellungen zur Lösung spezifischer Probleme geben. Für den Studierenden bietet sie eine Möglichkeit zur Wissensvertiefung. Sie bleibt damit unter erweitertem Namen und in der inzwischen dritten Generation in der bewährten Konzeption, die ihr die Vorgänger am ISW, Professor Gottfried Stute (1972 - 1982) und die Professor Günter Pritschow (1984 - 2005), gegeben haben.

Der vorliegende Band beschäftigt sich mit einem Informationsmodell, das sowohl die Beschreibung von Automatisierungsgeräten als auch die durch sie bereitgestellte Kommunikationsobjekte erlaubt. Auf diese Weise wird eine funktionale Sicht auf die Geräte und die funktionalen Zusammenhänge über Gerätegrenzen hinaus bereitgestellt. Hierfür werden zunächst bestehende Lösungen anhand einer in der Arbeit hergeleiteten Methode zum Vergleich von Beschreibungssprachen an den Anforderungen einer funktional orientierten Projektierung gespiegelt. Im Weiteren werden ein entsprechendes Logisches Modell und eine funktionale Klassifikation von Automatisierungsgeräten erarbeitet.

Der Herausgeber dankt der Druckerei für die drucktechnische Betreuung und dem Jost Jetter Verlag für die Aufnahme der Reihe in sein Lieferprogramm.

A. Verl

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) der Universität Stuttgart.

Herrn Prof. Dr.-Ing. Peter Klemm danke ich herzlich für die wohlwollende Unterstützung und die wertvollen Anregungen, die zum Gelingen dieser Arbeit wesentlich beigetragen haben, sowie für die Übernahme des Hauptberichtes.

Herrn Prof. Dr.-Ing. Dr. h.c. mult. Paul J. Kühn danke ich herzlich für seine Bereitschaft, den Mitbericht zu übernehmen.

Weiterhin danke ich allen aktuellen und ehemaligen Mitarbeitern des Institutes, vor allem im Fachbereich Steuerungstechnik, für die kollegiale und offene Zusammenarbeit und die interessanten Diskussionen. Insbesondere danke ich den Herren Dipl-.Ing. Christian Kircher, Dr.-Ing. Stefan Staudt und Dr.-Ing. Thomas Bürger für die kritische Durchsicht meiner Arbeit.

Nicht zuletzt danke ich meiner Frau Corina für ihre tatkräftige moralische Unterstützung während meines Studiums, der Arbeit am Institut und bei der Erstellung dieser Ausarbeitung.

Andreas Selig

Inhaltsverzeichnis

Abstract **16**

1 Einleitung **18**

2 Begriffsdefinitionen **20**
2.1 Automatisierungspyramide 20
2.2 Das OSI-Referenzmodell 22
2.3 Prinzip, Verfahren und Methode 25
2.4 Prozessmodell 26
2.5 Konfigurieren und Parametrieren 26
2.6 Engineering 26
2.7 Informationsmodell 27
2.8 Lebenszyklus 28
2.9 Gerät 29
2.10 Funktionalität und Funktion 29
2.11 Kommunikationsobjekt 30
2.12 Funktionale Gruppe 31
2.13 Profil 31
2.14 Konformität 33
2.15 Gerätemodell 33
2.16 Gerätebeschreibungssprache 33
2.17 Gerätebeschreibung 35
2.18 Objektorientierung 35
2.19 Geräteklasse 36
2.20 Subprofil und Superprofil 36
2.21 Typisierung 37

3 Anforderungen **38**
3.1 Ausgangssituation 38
3.2 Ziel der Arbeit 41
3.3 Anforderungen an ein Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten 43
3.4 Vorgehen in der Arbeit 43

3.4.1 Methode zum Vergleich von Beschreibungssprachen 44
3.4.2 Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten 45
3.4.3 Funktionale Klassifikation von Automatisierungsgeräten 45
3.4.4 Beschreibungsmöglichkeit für das Informationsmodell 46
3.5 Zusammenfassung 46

4 Stand der Technik 48
4.1 Gängige Dateiformate von Gerätebeschreibungsdateien 48
4.1.1 Initialisierungsdateiformat 48
4.1.2 XML-Dateiformat 49
4.2 Gerätebeschreibungssprachen 49
4.2.1 Generic Station Description und Generic Station Description Markup Language 50
4.2.2 Field Device Configuration Markup Language 50
4.2.3 Electronic Data Sheet 50
4.3 Herstellerübergreifende Mechanismen zur Integration von Geräten in ein Engineeringwerkzeug 51
4.3.1 Electronic Device Description Language 51
4.3.2 Field Device Tools/Device Type Manager 52
4.4 Formale Spezifikation von Schnittstellen 53
4.4.1 Mathematische Spezifikation 54
4.4.2 Axiomatische Spezifikation 54
4.4.3 Algorithmische Spezifikation 54
4.4.4 Spezifikation mittels Traces 55
4.4.5 Spezifikation mittels temporaler Logik 55
4.5 Graphentheorie 55
4.6 Zusammenfassung und Bewertung 56

5 Methode zum Vergleich von allgemeinen Beschreibungssprachen 58
5.1 Mächtigkeit einer Gerätebeschreibungssprache 58
5.2 Die Entstehung einer Gerätebeschreibungssprache 59
5.3 Ableiten der Methode zum Vergleich allgemeiner Beschreibungssprachen 61
5.3.1 Ausgangssituation 61
5.3.2 Bewertung von Methoden zum Vergleich von Beschreibungssprachen 61

5.4 Vergleich und Bewertung heutiger Sprachen 63
5.4.1 Generic Station Description und Generic Station Description Markup Language 63
5.4.2 Field Device Configuration Markup Language 65
5.4.3 Electronic Data Sheet 66
5.5 Zusammenfassung 68

6 Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten 70
6.1 Anforderungsanalyse 70
6.2 Konzeption 70
6.2.1 Definition eines allgemeingültigen Gerätemodells 72
6.2.2 Einordnen von Metaprofilen in das logische Modell 73
6.2.3 Aufbau des Gerätemodells 75
6.3 Zusammenfassung 76

7 Funktionale Klassifikation von Automatisierungsgeräten 77
7.1 Funktionale Profile für Automatisierungsgeräte 77
7.1.1 Notwendigkeit für funktionale Profile 77
7.1.2 Strukturierung von Profilen 77
7.2 Vorgehen zur funktionalen Klassifikation von Automatisierungsgeräten 78
7.3 Methode zum Festlegen vom Mengen von Kommunikationsobjekten 80
7.3.1 Fähigkeiten von Kommunikationsobjekten 80
7.3.2 Beziehungen zwischen Fähigkeiten von Kommunikationsobjekten 82
7.3.3 Anwendungsfälle für einen Abhängigkeitsgraphen von Kommunikationsobjekten 94
7.3.4 Regeln für die Gruppierbarkeit von Kommunikationsobjekten 95
7.3.5 Ableitung von Profilen und deren Aufbau aus der Gruppierung von Kommunikationsobjekten 97
7.4 Interaktionsgraph von Kommunikationsobjekten 99
7.5 Klassifikation von Automatisierungsgeräten 100
7.5.1 Funktionale Klassifikation von Automatisierungsgräten auf Kommunikationsebene 100
7.5.2 Funktionale Klassifikation von Automatisierungsgeräten auf Applikationsebene 102

7.6 Zusammenfassung 105

8 Beschreibungsmöglichkeit für die funktionale Typisierung von Automatisierungsgeräten im Informationsmodell 107
8.1 Informationsmodell für die Darstellung von Profilen 107
8.1.1 Klassendarstellung des Informationsmodells für die Darstellung von Profilen 107
8.1.2 Umsetzung in einem XML-Schema 116
8.2 Informationsmodell zur Darstellung von Geräten 116
8.2.1 Klassendarstellung des Informationsmodells für die Darstellung von Geräten 117
8.2.2 XML-Schema des Informationsmodells für die Darstellung von Geräten 122
8.3 Zusammenfassung 123

9 Realisierung 124
9.1 Methode zum Vergleich der Mächtigkeit von Beschreibungssprachen 124
9.2 Strukturierung von Profilen 126
9.3 SDDML und SPDML 129
9.4 Umsetzung in Engineeringwerkzeugen 129
9.5 Zusammenfassung 130

10 Zusammenfassung und Ausblick 131
10.1 Zusammenfassung 131
10.2 Ausblick 132

Literaturverzeichnis 133

Abkürzungen

CAN	Controller Area Network
CiA	CAN in Automation
CIP	Common Industrial Protocol
CoE	CANopen over EtherCAT
DDL	Device Description Language
DIN	Deutsches Institut für Normung
DTM	Device Type Manager
E/A	Eingang/Ausgang
EBNF	Erweiterte Backus-Naur-Form
ECT	EDDL Cooperation Team
EDDL	Electronic Device Description Language
EDS	Electronic Datasheet
ERP	Enterprise Resource Planing
FDCML	Field Device Conguration Markup Language
FDT	Field Device Tools
FTP	File Transfer Protocol
GSD	Generic Station Description

GSDML Generic Station Description Markup Language

GUI Graphical User Interface

HMI Human Machine Interface

HTTP Hypertext Transfer Protocol

IEC International Electrotechnical Commission

IP Internet Protocol

ISO International Organization for Standardization

MAU Medium Access Unit

MC Motion Control (Bewegungssteuerung)

MES Manufacturing Execution System (Leitsystem)

NC Numerical Control (Numerische Steuerung)

OLE Object Linking and Embedding

OPC OLE for Process Control

OSI Open Systems Interconnection

PNO Profibus Nutzer Organisation e.V.

Profibus Process Field Bus

Profinet Process Field net

RC Roboter Control (Robotersteuerung)

RFC Request for Comment

SCADA Supervisory Control and Data Acquisition

SDDML SERCOS Device Description Markup Language

SERCOS Serial Realtime Communication System

SPDML SERCOS Profile Description Markup Language

SPS Speicherprogrammierbare Steuerung

TCP Transmission Control Protocol

UDP User Datagram Protocol

UML Universal Modelling Language

XML Extensible Markup Language

XSLT Extensible Stylesheet Language Transformation

Abstract

As yet the so-called "plug and produce" technique in automation technology - which is an adaptation of the well-known "plug and play" technique - has not yet been fully achieved, even though it is continually in demand and often proposed. One of the main reasons for this is the complexity of the communication media used in these systems. Modern communication media implement complex and multilayered interfaces between the devices involved. These interfaces depend not only upon the communication medium but also upon the architecture of control and the degree of centralization employed.

In order to getting one step closer to the goal of “plug and produce”, an integrated information model for automation technology is needed. This model must offer a functional view of the devices involved and the communication objectives provided by these devices.

By comparing device description languages that exist in automation technology and their underlying information models, it turned out that none of these concepts has the ability to map functional connections and dependencies.

Therefore this thesis proposes an information model for functional typification of devices used in automation technology. In this model it is possible to describe functional dependencies across the boundaries between devices. It is also possible to provide a functional perspective upon these devices, including engineering tools, even when they are being used offline. This model can be used at a functional level to test whether the communication objectives needed by an application are enabled by the devices in question.

In addition, two methods have been developed that may also be of use outside the scope of this thesis. One method can be used to compare description languages by deriving the underlying information model. The description languages are then compared to the level of the model.

One benefit of this method is that it is possible to develop methods of translating the various description languages. In this way it is possible to construct the relationships between objects in different description languages and also find out what objects cannot be mapped.

The other method provides the facility of grouping communication objects into functional units. Using this method it is possible to derive all levels of implementation of the devices involved that are compliant with a given specification. This method even allows finding potential errors in specifications by revealing conflicts in the relationships between communication objects. This method is based on graph theory and matrix computing.

The methods and results of this thesis provide a basis for the creation of a specific information model of an Ethernet-based field bus. They also have an impact on the work of the technical working-groups and the specification of this field bus. In addition, the methods and results obtained have been implemented and thus proven in commercial engineering tools.

1 Einleitung

Maschinen und Anlagen sind heute sowohl mechanisch als auch steuerungstechnisch weltweit auf einem sehr hohen Niveau. Die grundlegenden Maschinenstrukturen sind seit Jahrzehnten bewährt. Ein Unternehmen kann sich deshalb nur dann auf dem Weltmarkt erfolgreich behaupten, wenn es über durchdachte Produktfamilien verfügt, die sich aus Baukastensystemen /1/ heraus zusammenstellen lassen.

Parallel zur Bestrebung, die Produktfamilien in Baukästen zu standardisieren, führt die zunehmende Miniaturisierung und die Verfügbarkeit von kostengünstigen Mikroprozessoren zur Integration zusammengehörender Funktionen. So entstehen mechatronische Module auf Basis einer Gerätehardware. Maschinen und Anlagen werden daher immer mehr durch Kombination von mehreren mechatronischen Modulen unter Berücksichtigung produktionsprozessspezifischen Bedingungen aus einem Baukasten heraus konfiguriert /2/.

Die schnelle kundenindividuelle Konfigurierung von Maschinen und Anlagen aus technologisch anspruchsvollen Baugruppen und die Integration dieser Komponenten zu technologisch spezialisierten Maschinen- und Anlagenkonfigurationen ist deshalb zu einem entscheidenden Wettbewerbsvorteil deutscher Maschinen- und Anlagenbauern gegenüber ausländischen Herstellern geworden. Diese können zwar kostengünstig produzieren, für die hohe funktionale Integration und Konfiguration komplexer Maschinen und Anlagen fehlt ihnen aber bis jetzt noch das Know-how deutscher Ingenieure und Facharbeiter. Das so genannte Engineering von Maschinen und Anlagen rückt dadurch immer stärker in den Vordergrund /3/, /4/.

Aktueller Trend in der Automatisierungstechnik ist die Einführung von Ethernet-basierten, digitalen Schnittstellen zwischen der Steuerungs- und der Sensor-/Aktorebene. Die entstehenden Ethernet-basierten Feldbusse decken dabei nicht nur die klassischen Feldbusaufgaben ab, sondern bieten zusätzlich Bandbreite für weitere Dienste /5/. Ziel eines jeden Ethernet-basierten Feldbusses ist dabei, ein umfassendes Kommunikationsmedium für die Automatisierungstechnik zu sein /6/. Dieses soll den Anforderungen eines Feld-, Antriebs- und Sicherheitsbusses gerecht werden und zusätzlich noch Kommunikation über Protokolle, die aus der elektronischen Datenverarbeitungstechnik bekannt sind, ermöglichen, um eine vertikale Integration in höherliegende Ebenen der Automatisierungspyramide zu erlauben /7/.

Ein „Plug and Produce“, ähnlich dem aus der Computertechnik bekannten „Plug and Play“, bei dem sich die beteiligten Komponenten gegenseitig erkennen und sich das Gesamtsystem automatisch konfiguriert und parametriert, wird in diesem Umfeld allenthalben gefordert und propagiert /8/, /9/, da die Komplexität und der Umfang der Schnittstellen zwischen den beteiligten Geräten immer größer und umfangreicher werden /10/, /11/. Es gibt bislang kein globales Konzept für die Automatisierungstechnik, dieses Problem zu lösen /12/.

Die vorliegende Arbeit soll einen Beitrag zur Erreichung dieses Ziels darstellen, indem sie ein Informationsmodell für Automatisierungsgeräte und die durch sie bereitgestellten Kommunikationsobjekte liefert. Auf diese Weise wird eine funktionale Sicht auf die Geräte und die funktionalen Zusammenhänge über Gerätegrenzen hinaus bereitgestellt.

Um dieses Ziel zu erreichen, werden zunächst bestehende Informationsmodelle aus der Automatisierungstechnik aus marktgängigen Gerätebeschreibungssprachen abgeleitet Diese Informationsmodelle werden dann den Anforderungen des „Plug and Produce“ gegenübergestellt. Um die Informationsmodelle zu vergleichen, wird zunächst eine Methode zum Vergleich von Informationsmodellen hergeleitet.

Aufbauend auf der Anforderungsanalyse werden Informationsmodelle für die Darstellung von Geräten und den durch sie bereitgestellten Kommunikationsobjekten erarbeitet. Es werden Methoden erarbeitet, die dazu dienen, Graphen funktionaler Abhängigkeiten zwischen Kommunikationsobjekten aus einer Spezifikation abzuleiten.

Diese Graphen erlauben einerseits die funktionale Gruppierung der Kommunikationsobjekte, andererseits ermöglichen sie es, Geräte funktional zu klassifizieren. Darüber hinaus können aus den Graphen Fehler oder Unstimmigkeiten innerhalb einer bereitgestellten Spezifikation aufgedeckt werden.

Auf diese Weise kann der Vorgang des „Plug and Produce“ unterstützt werden, da die erarbeiteten Zusammenhänge informationstechnisch durch ein Engineeringwerkzeug ausgewertet werden können. Diese Informationen können so dem Nutzer bei der Auswahl der benötigten Geräte bereitgestellt werden. Dies erleichtert die Konfigurierung von individuellen Maschinen oder Anlagen aus Baukästen heraus, reduziert Fehler und den notwendigen Engineering-Aufwand.

1 Begriffsdefinitionen

In diesem Kapitel werden die zum Verständnis der vorliegenden Arbeit notwendigen Begriffe definiert. Für viele von diesen existieren unterschiedliche Begriffserklärungen sowohl in unterschiedlichen Disziplinen als auch innerhalb der Automatisierungstechnik. Im Rahmen der Arbeit werden sie in dem in diesem Kapitel definierten Sinn verstanden.

1.1 Automatisierungspyramide

Die **Automatisierungspyramide** ist in Bild 2.1 dargestellt /13/. Sie besteht aus fünf Ebenen:

- Sensor-/Aktorebene
- Steuerungsebene
- Prozessleitebene
- Betriebsleitebene
- Unternehmensleitebene

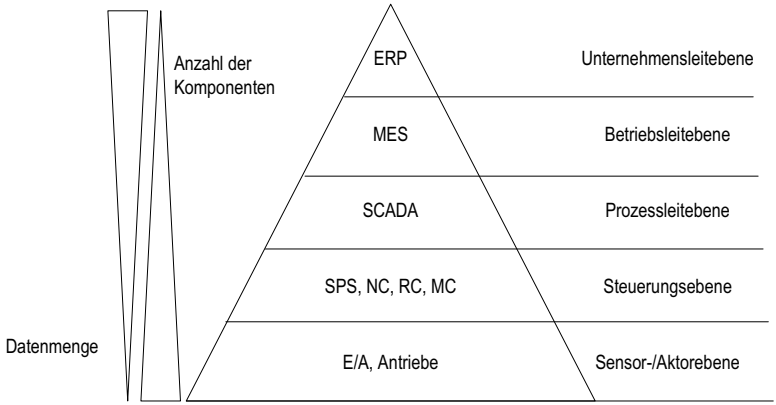
Auf der **Sensor-/Aktorebene** befinden sich die Feldgeräte. Auf dieser Ebene werden mittels Sensoren die Prozessgrößen gemessen und durch die Aktoren auf diese eingewirkt.

In der **Steuerungsebene** werden die in der Sensor-/Aktorebene durch Sensoren gemessenen Prozessgrößen verarbeitet und durch Steuer- und Regelalgorithmen zu Stellgrößen für die Aktoren verarbeitet.

Auf der **Prozessleitebene** werden alle Aufgaben zur Führung, Planung und Koordination eines aus mehreren Steuerungen bestehenden technischen Systems zusammengefasst.

Die **Betriebsleitebene** ist die Ebene der Leitrechner. Hier werden alle Aufgaben zur Führung, Planung und Koordination eines aus mehreren Teilsystemen bestehenden Automatisierungssystems wie einer Fertigungsstraße durchgeführt.

In der **Unternehmensleitebene** werden alle zur Führung einer Fabrik oder eines Unternehmens notwendigen langfristigen Planungen und Vorgaben erarbeitet und an die Betriebsleitebene weitergeleitet.



E/A	Eingang/Ausgang
SPS	Speicherprogrammierbare Steuerung
NC	Numerical Control
RC	Roboter Control
MC	Motion Control
SCADA	Supervisory Control and Data Acquisition
MES	Manufacturing Execution System
ERP	Enterprise Resource Planing

Bild 2.1: Automatisierungspyramide

2.2 Das OSI-Referenzmodell

Das **OSI (Open System Interconnection)-Referenzmodell** /14/ ist eine von der ISO definierte Designgrundlage für Kommunikationsprotokolle. In ihm werden die Aufgaben der Kommunikation zwischen zwei Systemen in sieben unterschiedliche Schichten aufgeteilt. Bild 2.2 zeigt die Schichten, den Datenfluss und die virtuellen Verbindungen auf Schichtenebene.

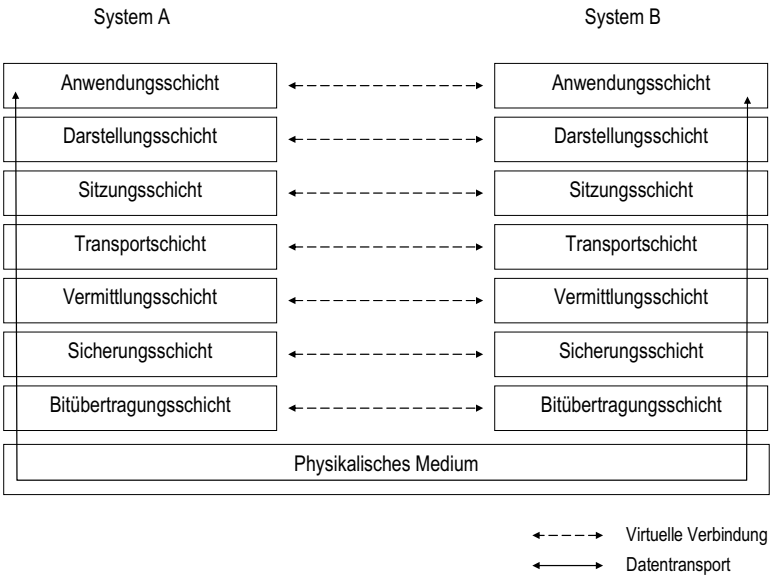


Bild 2.2: Das OSI-Referenzmodell

Das OSI-Referenzmodell besteht aus folgenden sieben Schichten:

- Bitübertragungsschicht
- Sicherungsschicht
- Vermittlungsschicht
- Transportschicht
- Sitzungsschicht

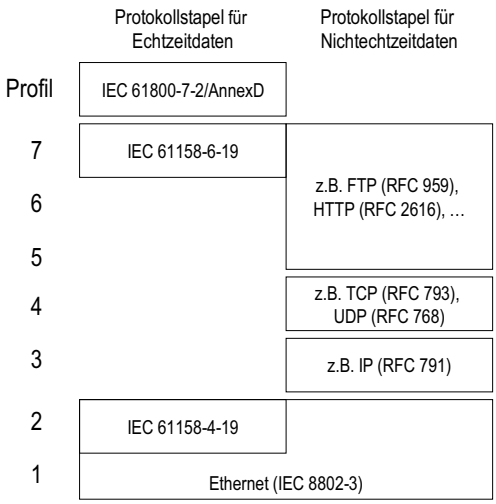
- Darstellungsschicht
- Anwendungsschicht

Für die in der Automatisierungstechnik zum Datenaustausch zwischen der Sensor-/Aktorebene und der Steuerungsebene verwendeten Feld- und Antriebsbusse sind von diesen Schichten meist nur die Anwendungsschicht, die Sicherungsschicht und die Bitübertragungsschicht relevant /15/, /16/.

Bild 2.3 zeigt am Beispiel eines SERCOS III-Antriebsgerätes den in einem Automatisierungsgerät implementierten Protokollstapel, die jeweilige relevante Norm beziehungsweise den RFC, auf dem die Implementierung beruht und die Einordnung in das OSI-Referenzmodell.

Der linke Teil stellt den Protokollstapel für die Echtzeitdaten dar. SERCOS III erweitert die Ethernet-Norm IEC 8802-3 /17/ in der Sicherungsschicht um Echtzeitfähigkeit im für SERCOS III relevanten Teil der IEC 61158-4 /18/. Auf der Applikationsschicht definiert SERCOS III in dem relevanten Teil der IEC 61158-6 /19/ ein Protokoll zum Datenaustausch. Darüber hinaus wird ein der IEC 61800-7-2 /20/ im Annex D eine Semantik der Datenobjekte definiert.

Neben dem Protokollstapel für Echtzeitdaten existiert optional in einem SERCOS III-Antriebsgerät auch ein Protokollstapel für Nichtechtzeitdaten. Dieser kann zum Beispiel die in Bild 2.3 auf der rechten Seite dargestellten Protokolle IP /21/, TCP /22/, UDP /23/, FTP /24/ und HTTP /25/ beinhalten.



TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
RFC	Request for Comment
IEC	International Electrotechnical Commission

Bild 2.3: Protokollstapel in einem SERCOS III Antriebsgerät /26/

Die **Bitübertragungsschicht** ist die unterste Schicht innerhalb des OSI-Referenzmodells. Die Aufgabe dieser Schicht ist es, physikalische Verbindungen auf- und wieder abzubauen, diese aufrechtzuerhalten und Bits entsprechend des physikalischen Mediums zu übertragen.

Aufgabe der **Sicherungsschicht** ist es, eine fehlerfreie Übertragung der zu sendenden Bits zu gewährleisten. Dies geschieht beispielsweise durch das Hinzufügen von Prüfsummen und Folgenummern. Fehlerhafte, verfälschte oder verloren gegangene Segmente können vom Empfänger erneut angefordert werden.

Die **Vermittlungsschicht** sorgt dafür, dass die Daten beim richtigen Empfänger ankommen. Dies schließt eine Wegesuche zwischen den Sender und Empfänger mit ein. Da nicht immer eine direkte physikalische Kommunikation zwischen Sender und

Empfänger möglich ist, kann es vorkommen, dass Knoten, die zwischen Sender und Empfänger liegen, die Daten weiterleiten müssen. In diesen dazwischen liegenden Knoten werden die Daten nur bis zur Verbindungsschicht interpretiert und dann weitergeschickt.

Die **Transportschicht** ist die unterste Schicht, die eine vollständige Ende-zu-Ende Kommunikation zwischen Sender und Empfänger zur Verfügung stellt. Sie bietet den darüber liegenden Schichten einen einheitlichen Zugriff, so dass diese keine Kenntnis des darunter liegenden Kommunikationsnetzwerkes haben müssen.

Die **Sitzungsschicht** sorgt für die Kommunikation zwischen Prozessen auf zwei Systemen. Aufgabe der **Darstellungsschicht** ist die systemabhängige Darstellung der Daten in eine unabhängige Form und ermöglicht somit den syntaktisch korrekten Datenaustausch zwischen unterschiedlichen Systemen. Dazu gehören auch die Datenkompression und die Datenverschlüsselung.

Die **Anwendungsschicht** ist die oberste der sieben Schichten des OSI-Referenzmodells. Sie stellt den Anwendungen eine Vielzahl an Funktionalitäten zur Verfügung. Der eigentliche Anwendungsprozess liegt oberhalb der Schicht und wird vom OSI-Referenzmodell nicht beschrieben.

2.3 Prinzip, Verfahren und Methode

Die Begriffe Prinzip, Verfahren und Methode definiert /27/. Demnach wird unter einem **Prinzip** der Grundsatz, der einer Handlung zugrunde liegt, verstanden und somit die theoretische Grundlage für ein Vorgehen. Ein **Verfahren** beschreibt konkrete Wege zum Lösen eines Problems oder einer Gruppe von Problemen.

Unter **Methode** wird eine angewandte und begründete Vorgehensweise zum Erreichen eines oder mehrerer festgelegter Ziele unter Berücksichtigung festgelegter Prinzipien verstanden. Methoden helfen somit, Prinzipien anwendbar zu machen. Eine Methode ist dabei konkret, das bedeutet, dass sie den Weg zum Erreichen des Ziels in eine Abfolge verschiedener Verfahren aufteilt.

2.4 Prozessmodell

Der Begriff **Prozessmodell** entstammt der Informatik und beschreibt ein für die Softwareentwicklung angepasstes Vorgehensmodell. Das Prozessmodell legt die Sequenz der Arbeitsschritte fest. Jedem Arbeitsschritt ist hierbei eine Person oder Personengruppe zugeordnet, die diesen ausführt. Zusätzlich besitzt jeder Arbeitsschritt ein definiertes Ergebnis und eine Überprüfungsvorschrift durch das Qualitätsmanagement /27/.

2.5 Konfigurieren und Parametrieren

Die Begriffe **Konfigurieren** und **Parametrieren** beschreibt die DIN 19226 in Teil 5 /28/. Unter Konfigurieren wird demnach das Zusammensetzen von Regelungs- und Steuerungssystemen aus vorgegebenen Funktions- oder Baueinheiten verstanden. Parametrieren bedeutet das Zuweisen von Werten von bestimmten Parametern zum Einstellen eines gewünschten Verhaltens. In Anlehnung an diese Definition wird im Rahmen der Arbeit unter Konfigurieren der Vorgang des Einstellens der Kommunikationseigenschaften verstanden werden. Unter Parametrieren wird im Rahmen der Arbeit das Einstellen für die Applikation wichtigen Eigenschaften verstanden.

2.6 Engineering

Engineering ist ein interdisziplinärer Ansatz, um Systeme zu entwickeln und zu realisieren. Im Rahmen der Automatisierungstechnik werden unter diesem Ausdruck ingenieurstechnische Arbeiten verstanden, die im Laufe des Produktlebenszyklus einer Maschine oder Anlage anfallen /29/.

Diese Arbeiten werden heute meist durch ein Engineeringwerkzeug unterstützt /30/. Darunter fallen Dinge wie das Planen der Anlage, das Konfigurieren und Parametrieren der einzelnen Komponenten, das Entwickeln des Steuerungsprogramms und die Betreuung und Auswertung der Daten im laufenden Betrieb.

Für die automatisierte Unterstützung der Engineeringaufgaben durch ein Werkzeug sind im Engineeringwerkzeug Regeln definiert. Diese werden meist auf Grundlage eines softwaretechnischen Abbildens der Realität in Form eines Informationsmodells definiert.

2.7 Informationsmodell

In einem **Informationsmodell** wird die Summe aller für den Anwendungsfall relevanten Informationen der betrachteten Objekte und Sachverhalte zusammengefasst /27/. Die Auswahl der darzustellenden Informationen und deren Strukturierung hängen von dem Anwendungsfokus des Informationsmodells ab.

Um ein Informationsmodell zu erhalten, werden die in Bild 2.4 dargestellten Schritte durchlaufen. Zunächst wird durch Abstraktion des relevanten Ausschnitts aus der Realität ein Mentales Modell abgeleitet. Dieses führt dann durch die Formalisierung zum Informationsmodell. Dieses Vorgehen wird Informationsmodellierung genannt.

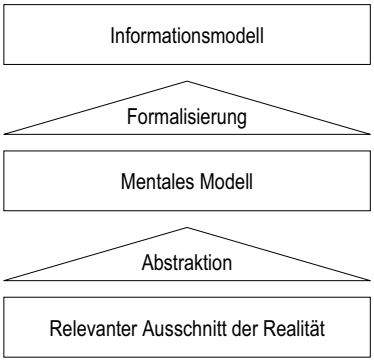


Bild 2.4: Informationsmodellierung /27/

Zur Strukturierung von Informationen im Informationsmodell werden Ontologien verwendetet. Der Begriff **Ontologie** stammt ursprünglich aus der Philosophie, und beschreibt dort den Aspekt der Metaphysik, der sich mit dem Wesen des Seins und des Seienden befasst /31/. Übertragen in die Informatik ist **Ontologie** ein Konzept zur Strukturierung von Information /32/.

2.8 Lebenszyklus

Der **Lebenszyklus** eines Produktes wird in der Norm ISO 9004 /33/ definiert. Er besteht im Allgemeinen aus zwölf Phasen, wie es in Bild 2.5 dargestellt ist.

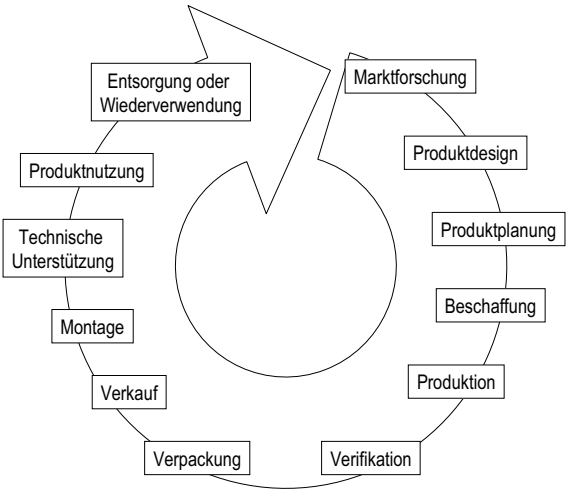


Bild 2.5: Produktlebenszyklus nach ISO 9004

Für Automatisierungsanlagen sind nicht alle in der Norm vorgeschlagenen Phasen relevant. Es ergeben sich für Anlagen folgende Phasen /34/:

- Produktidee
- Konzepterstellung
- Entwicklung
- Konstruktion
- Prototypenphase
- Produktion
- Instandhaltung
- Entsorgung bzw. Wiederverwendung

2.9 Gerät

Unter **Gerät** wird in der Automatisierungstechnik eine abgeschlossene Komponente verstanden /35/. 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.

2.10 Funktionalität und Funktion

In einem Gerät befinden sich **Funktionalitäten**. Hierbei handelt es sich um funktional abgeschlossene Methoden zur Bearbeitung von Aufgaben. Teile dieser Funktionalitäten werden über externe Schnittstellen wie beispielsweise ein Bussystem anderen Geräten zur Verfügung gestellt. Funktionalitäten, die so durch eine Kommunikationsschnittstelle nach außen bereitgestellt werden, werden **Funktionen** genannt (Bild 2.6).

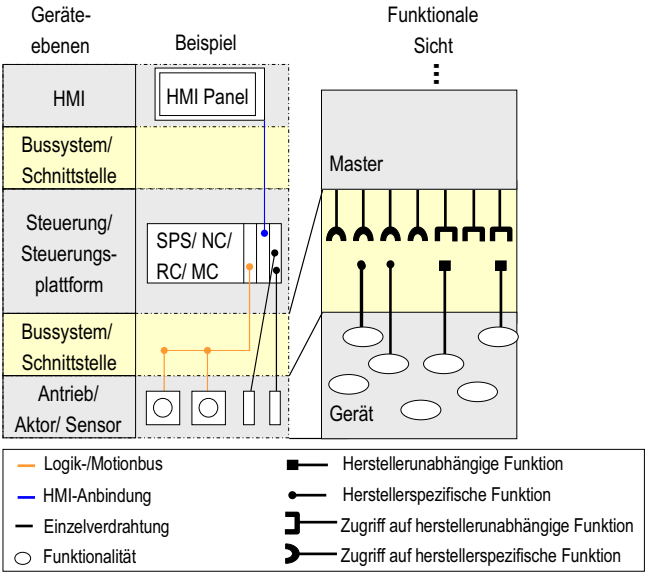


Bild 2.6: Darstellung von Funktion und Funktionalität

Nicht nach außen bereitgestellte Funktionalitäten sind nur innerhalb des Gerätes genutzte softwaretechnisch implementierte Methoden. Diese können aber durch Funktionen innerhalb des Gerätes benutzt werden.

Es gibt dabei verschiedene Zugriffsarten auf diese Funktionen. Sind die Funktionen und der Zugriff auf sie in einem herstellerübergreifenden Standard genormt, so spricht man von herstellerunabhängigen Funktionen.

Sind die Funktionen nicht herstellerübergreifend genormt, oder ist der Zugriff auf sie nicht herstellerübergreifend spezifiziert, so spricht man von proprietären Funktionen.

Auf die Funktionen wird von anderen Geräten über entsprechend genormte oder proprietäre Mechanismen unter Benutzung einer gemeinsamen Kommunikationsschnittstelle zugegriffen.

2.11 Kommunikationsobjekt

Unter einem **Kommunikationsobjekt** wird ein Objekt innerhalb eines Gerätes verstanden, welches nach außen für andere Kommunikationsteilnehmer zugreifbar ist. Ein solches Kommunikationsobjekt kann in der Automatisierungstechnik entweder zyklisch, azyklisch oder beides sein, wie es in Bild 2.7 dargestellt ist /36/. Ein Kommunikationsobjekt kann entweder eine Funktion des Gerätes oder Daten aus der geräteinternen Datenhaltung kapseln.

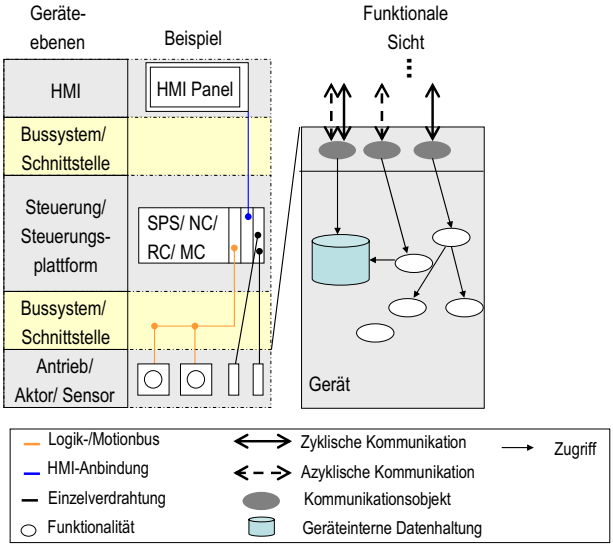


Bild 2.7: Definition des Begriffes Kommunikationsobjekt

2.12 Funktionale Gruppe

Eine **Funktionale Gruppe** fasst unterschiedliche Kommunikationsobjekte, die zusammen eine definierte abgeschlossene Gruppe an Eigenschaften und Funktionen innerhalb eines Gerätes bündelt, zu einer Gruppe zusammen.

Ein Beispiel für eine Funktionale Gruppe ist die Gruppe aller Funktionen, die sich mit dem Ablauf dem Herstellen eines absoluten Lagebezugs eines Antriebsgerätes befassen, wie Auslesen des Lage-Istwertes, Erkennen eines Referenzpunktes und Anpassen des internen Lagebezugs durch Setzen eines neuen Bezugswertes des Lagesystems im Gerät.

2.13 Profil

Der Begriff **Profil** wird in Bild 2.8 dargestellt. Ein Profil definiert sich als die Summe mehrerer Funktionaler Gruppen /37/.

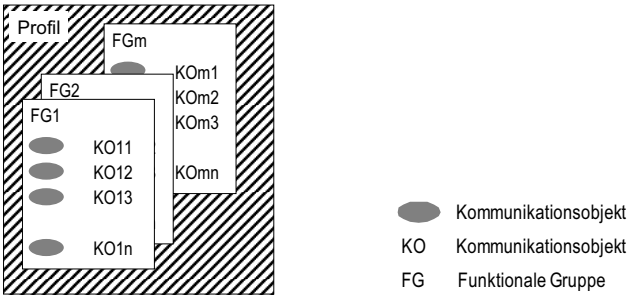


Bild 2.8: Profil

Es existieren zwei Arten von Profilen in der Automatisierungstechnik: Die **Kommunikations-** und die **Applikationsprofile**. Kommunikationsprofile definieren sich ausschließlich durch die Beschreibung von Funktionalen Gruppen, die die reine Kommunikation zwischen Geräten beschreiben. Sie legen somit fest, ob und wie einzelne Geräte miteinander Daten austauschen können. Es können somit unabhängig von der Geräteklasse die Kommunikationseigenschaften eines Bussystems beschrieben werden. Auf den Kommunikationsprofilen setzen Applikationsprofile auf. Diese beschreiben unabhängig vom verwendeten Kommunikationsmedium den Inhalt der ausgetauschten Daten.

Erst die Verwendung von Applikations- und Kommunikationsprofilen ermöglicht die komplette Beschreibung sowohl der Mechanismen der Kommunikation als auch der Dateninhalte. Die verbreiteten Applikationsprofile für elektrische Antriebe sind in der Norm IEC 61800 Part 7-2 zusammengefasst /20/. Diese Norm beinhaltet die Beschreibung des CiA 402 Antriebsprofils, des CIP Motion-Profils, des PROFIdrive-Profils und des SERCOS-Antriebsprofils.

Das Verwenden dieser Applikationsprofile mit den entsprechenden Kommunikations-profilen beschreibt Part 7-3 der Norm 61800 /38/.

2.14 Konformität

Durch **Konformität** wird sichergestellt, dass ein Gerät sich in Übereinstimmung mit einem Standard oder einer Norm verhält. Dies beinhaltet das komplette Verhalten des Gerätes, also sowohl die Reaktion auf eine Anfrage als auch die Generierung einer solchen /39/, /40/.

2.15 Gerätemodell

Ein **Gerätemodell** ist ein Informationsmodell für Automatisierungsgeräte und definiert die gemeinsamen Struktureigenschaften einer betrachteten Menge von Geräten innerhalb eines Automatisierungssystems /41/. Die konkrete Ausprägung des Gerätemodells ist abhängig davon, in welcher Phase des Lebenszyklus einer Automatisierungsanlage und auf welcher Ebene der Automatisierungspyramide das Gerätemodell zum Einsatz kommt. Es legt somit fest, wie die logische und funktionale Sicht auf ein Gerät in dem beschriebenen Kontext ist.

2.16 Gerätebeschreibungssprache

Eine **Gerätebeschreibungssprache** ist die formalisierte Darstellung eines Gerätemodells. Der logische Aufbau des Gerätemodells legt die notwendigen somit Beschreibungselemente für eine Gerätebeschreibungssprache fest /42/.

Eine Sprache wird definiert durch eine Menge an vorhandenen Worten, die über einem zugrunde liegenden Alphabet gebildet werden können. Das Alphabet legt den Vorrat an Zeichen fest, die in der Sprache verwendet werden können.

Eine Sprache im Allgemeinen wird durch Syntax und Semantik definiert /43/. Unter Syntax werden Muster und Regeln verstanden, nach denen die definierten Zeichen zu Einheiten zusammengesetzt werden. Die Semantik legt die Bedeutung des Inhalts der Sprache fest und legt somit die Verbindung zwischen der Sprache und den zu beschreibenden Objekten der Realität fest.

Es existieren unterschiedliche Stufen der Formalisierung einer Sprache /44/. Dieser Grad der Formalisierung hängt vom Grad der Formalisierung von Syntax und Semantik der Sprache ab:

- Umgangssprache
- Fachsprache
- Sprache mit formal definierter Syntax
- Sprache mit formal definierter Syntax und formal definierter Semantik

Eine formale Sprache S über ein Alphabet A mit dem Wortvorrat A* ist folgendermaßen definiert, wobei sich der Wortvorrat A* durch die Aneinanderreihung einer endlichen Anzahl von Buchstaben aus A ergibt:

$S \subseteq A^*$ **und es existiert für S eine Definitionsvorschrift.**

Eine Sprache wird somit dadurch formal, dass eine formale Definition für die Syntax existiert, sie benötigt nicht unbedingt eine formale Definition für die Semantik. Eine solche Definition kann beispielsweise in der Erweiterten Backus-Naur-Form (EBNF) geschehen, wie sie in der Norm ISO 14977 /45/ beschrieben ist.

Beispiele für gängige Gerätebeschreibungssprachen in der Automatisierungstechnik sind:

- GSD (Generic Station Description) bzw. GSDML (Generic Station Description Markup Language) zur Beschreibung von Profibus- und Profinet-Geräten /46/,
- EDS (Electronic Datasheet) zur Beschreibung von Geräten, die ein CAN-basiertes Kommunikationsprotokoll wie DeviceNet und CANopen implementieren /47/,
- FDCML (Field Device Configuration Markup Language) zur Beschreibung von Interbus- /48/ und Drivecom-Geräten /49/.

Die ISO 15745 definiert ein Rahmenwerk zur Integration von Applikationen in offenen Systemen. Sie besteht aus fünf Teilen. In den Teilen 3 /50/ und 4 /51/ legt sie die grundlegende Struktur einer XML-basierten Beschreibung für Automatisierungsgeräte fest. Diese besteht aus folgenden Teilen:

- „DeviceIdentity“
- „DeviceFunction“
- „ApplicationLayers“
- „NetworkManagement-Layer“

Neuere Gerätebeschreibungssprachen wie GSDML und FDCML halten sich an diese grundlegende Struktur. Die oberste Strukturierungsebene der Beschreibung wird von der Norm vorgegeben, sieht also dort genau gleich aus. Sie unterscheiden sich aber ab der nächsten Ebene komplett, wo die den Beschreibungssprachen zugrunde liegenden Informationsmodelle abgebildet sind.

2.17 Gerätebeschreibung

Eine **Gerätebeschreibung** wird dazu verwendet, ein Gerät offline für die Konfigurierung und Parametrierung bereitzustellen /52/. Bei einer Gerätebeschreibungsdatei handelt es sich um eine Datei, die der Syntax und der Semantik einer Gerätebeschreibungssprache folgt.

Gerätebeschreibungsdateien werden hierfür dem Parametrierungs- und Konfigurierungswerkzeug bereitgestellt. Dieses liest aus den Dateien die Informationen heraus, die es zur Parametrierung und Konfigurierung braucht. Somit ist es möglich, ohne tatsächlich vorhandene Geräte offline die Schritte durchzuführen, die zur Parametrierung und Konfigurierung notwendig sind.

Ein weiterer Vorteil, der durch die Benutzung von Gerätebeschreibungen entsteht, ist, dass die beschriebenen Geräte kein absolutes Eigenbild brauchen. Das heißt, sie müssen nicht selber über alle ihre Möglichkeiten Bescheid wissen. Auf diese Weise müssen nicht alle Parameter und Eigenschaften eines Gerätes im Gerät abgelegt werden, was Speicher im Gerät spart.

2.18 Objektorientierung

Bei **Objektorientierung** handelt es sich um ein Paradigma aus der Informatik /53/. Die zu Grunde liegende Idee der Objektorientierung ist das Abbilden von Objekten der

realen Welt in Objekte eines Programms. Ein solches Objekt besitzt eine Menge von Eigenschaften und Verhaltensweisen.

Objekte mit gleichen oder ähnlichen Eigenschaften und Verhaltensweisen werden zu so genannten Klassen zusammengefasst und bilden dann Instanzen dieser Klasse. Haben unterschiedliche Klassen Ähnlichkeiten, können diese in einer Oberklasse zusammengefasst werden. Diese Oberklasse gibt ihren Unterklassen die Gemeinsamkeiten durch Vererbung weiter.

Eine weitere wichtige Eigenschaft der Objektorientierung ist die Kapselung. Dies bedeutet, dass ein Objekt auf die Eigenschaften eines anderen Objektes ausschließlich über den Aufruf einer Methode zugreifen kann.

2.19 Geräteklasse

Eine **Geräteklasse** ist eine Klasse von Geräten und fasst somit Geräte zusammen, denen eine Summe von Merkmalen oder Eigenschaften gemeinsam ist /54/. Dies kann beispielsweise dadurch geschehen, dass alle Geräte innerhalb einer Geräteklasse dasselbe spezielle Applikationsprofil implementieren. Dementsprechend bilden Antriebe eine Geräteklasse innerhalb der Automatisierungstechnik, da sie eine Summe von gemeinsamen Funktionen anbieten.

2.20 Subprofil und Superprofil

Ein **Subprofil** entspricht der Summe an Merkmalen oder Eigenschaften, die einer speziellen Geräteklasse gemein sind /55/. Antriebe im Allgemeinen implementieren ein Antriebsprofil. Eine abgeleitete Geräteklasse, wie Antriebe mit Lagesollwertschnittstelle implementiert ein Subprofil des Antriebsprofils. Umgekehrt ist ein **Superprofil** eine Verallgemeinerung mehrerer Profile oder Subprofile.

In einem Subprofil sind ausschließlich Kommunikationsobjekte vorhanden, die auch im Superprofil, dem Profil, von dem das Subprofil abgeleitet ist, vorhanden sind. Allerdings können die Eigenschaften der Funktionsgruppe innerhalb eines Subprofils andere sein.

Es können in einem Subprofil Funktionsgruppen verbindlich sein, die im Superprofil oder anderen Subprofilen nur optional sind.

2.21 Typisierung

Der Begriff **Typisierung** beschreibt das Zuordnen eines Objektes zu einer bestimmten Klasse. Im Rahmen der Arbeit ist unter Typisierung das Zuordnen eines Gerätes zu einer bestimmten Geräteklasse zu verstehen.

3 Anforderungen

3.1 Ausgangssituation

Das Dilemma der Rationalisierung ist es, die hohe Produktivität einer Großserie und Flexibilität einer Einzelteilfertigung in der Produktion zu vereinen, obwohl sich dies zunächst ausschließt /56/.

Die Forderung nach hoher Variantenzahl und der Möglichkeit, auch geringe Losgrößen kostengünstig und mit hoher Qualität zu produzieren, fordert eine hohe Flexibilität der gesamten Wertschöpfungskette. Flexibilität bedeutet dabei, dass es möglichst schnell und einfach ist, die Produktion geänderten Randbedingungen wie Änderung der Nachfrage oder neue Aufgaben anzupassen.

Leitziele der gesamten Wertschöpfungskette sind dabei immer Kundenorientierung, Flexibilität, Qualität und aber auch Kostenersparnis.

Eine hohe Flexibilität von Maschinen und Anlagen werden heute dadurch erreicht, dass diese heute nicht mehr komplett neu erdacht werden, sondern sich aus Baukästen konfigurieren lassen /57/.

Die Verwendung von Baukastensystemen und die durch die zunehmende Miniaturisierung einhergehende zunehmende Komplexität der Funktionen von einzelnen Maschinenmodulen und Automatisierungsgeräten führen dazu, dass der Vorgang der Konfiguration und Rekonfiguration von Maschinen und Anlagen immer anspruchsvoller und damit zeitintensiver wird. Dem gegenüber steht die Forderung nach Kostenoptimierung durch schnellere Inbetriebnahme und geringere Stillstandszeiten bei der Rekonfiguration der Maschine.

Die Konfiguration und Rekonfiguration der Maschine oder Anlage und die Parametrierung der einzelnen Automatisierungsgeräte und mechantronischer Module werden immer stärker von Engineeringwerkzeugen unterstützt.

In der Automatisierungstechnik existieren heute vielfältige unterschiedliche Beschreibungssprachen und Mechanismen, um Automatisierungsgeräte oder mechatronische

Module in den Konfigurierungs- und Parametrierungsvorgang eines Engineering-werkzeugs zu integrieren. Diese sind in unterschiedlichen Phasen des Lebenszyklus der Automatisierungsanlage angesiedelt. Außerdem sind sie auf verschiedenen Ebenen innerhalb der Automatisierungspyramide zu finden.

Um die vertikale Integration von Sensoren und Aktoren voranzutreiben, werden aktuell zunehmend Ethernet-basierte, digitale Schnittstellen zwischen Steuerungs- und Sensor/-Aktorebene entwickelt und eingesetzt. Die Vorteile von Ethernet-basierten Lösungen auch auf Sensor-/Aktorebene sind dabei die verhältnismäßig große Bandbreite, die niedrigeren Kosten der Anschaltung und die Möglichkeit, der direkten Einbindung der Sensor-/Aktorebene in die vorhandenen Kommunikationsnetzwerke oberhalb der Steuerungsebene.

Die Ethernet-basierten digitalen Schnittstellen zielen darauf ab, die klassischen de-dizierten Feld-, Antriebs- und Sicherheitsbusse auf der Sensor-/Aktorebene abzulösen und als universales Netzwerk für die Automation zu dienen. Damit einhergehend werden die Schnittstellen, die zwischen den beteiligten Geräten bereitgestellt werden, zunehmend komplexer und vielschichtiger. Die Ausprägung dieser Schnittstellen hängt neben dem Kommunikationsmedium auch von den verwendeten Profilen und damit von der Systemarchitektur ab, also wie zentral oder dezentral die zugrunde liegende Steuerungsarchitektur aufgebaut ist. Die Automatisierungstechnik ist heute also geprägt von der Aufhebung der Trennung von Feld- und Antriebsbus.

Damit einhergehend entstehen neue Gerätestrukturen, die ebenfalls der Aufhebung der klassischen Trennung von Feld- und Antriebsbus Rechnung tragen. Einerseits entstehen zunehmend verteilte Systeme wie beispielsweise Antriebe mit getrennten Steuer- und Leistungsteilen und Ankopplung des Gebersystems über ein angeschlossenes Netzwerk.

Auf der anderen Seite vereinigen Hybridgeräte unterschiedliche Funktionen, wie Steuerungs-, Antriebs- und E/A-Funktionen in einem Gerät. Je nach Anforderung der Applikation können dabei beide beschriebenen Gerätestrukturen auch gemeinsam in einem Netzwerk auftreten.

Durch die klassische Trennung zwischen Antriebs- und Feldbus gab es in der Vergangenheit eine klare Aufgabenverteilung zwischen den beiden Bustypen. So wurden Servoantriebe für Anwendungen, in denen mehrere Achsen synchronisiert

Bahnen interpolieren, über echtzeitfähige Antriebsbusse wie SERCOS Interface an die Bewegungssteuerung angebunden.

Die Vernetzung dezentraler E/A-Peripheriegeräte mit der Logiksteuerung war bei einer solchen Struktur hingegen die Aufgabe eines klassischen Feldbusses wie Profibus.

Diese klare Aufgabentrennung und Anforderungen aus der Programmierung und Inbetriebnahme führen zu sehr unterschiedlichen Vorgehensweisen im Bezug auf die Konfiguration und Parametrierung. So besteht bei den dedizierten Antriebsbussen typischerweise keine Notwendigkeit für eine Konfiguration oder Parametrierung ohne wirklich vorhandene Maschine oder Anlage.

Die in der Maschine oder Anlage verbauten Servoantriebe werden während des Hochlaufs der Kommunikation durch die Steuerung automatisch konfiguriert und parametriert, die Feinabstimmung direkt bei der Inbetriebnahme vorgenommen. Neben der Auswahl der Betriebsart erfolgt hier zudem die Konfiguration der Prozessdaten, die nach dem Hochlauf im zyklischen Betrieb zwischen der Steuerung und den Antrieben ausgetauscht werden sollen.

Im Unterschied dazu ist die Anforderung bei den klassischen Feldbussen eine andere. Vor der Inbetriebnahme und ohne Vorhandensein aller Komponenten muss bereits im Vorfeld ein meist umfangreiches Programm geschrieben, optimiert und vorgetestet werden. Um dies zu können, muss die Abbildung der Eingangs- und Ausgangsdaten der E/A-Peripherie auf dem Bus bekannt sein. Dafür sorgt ein Buskonfigurator, der die von ihm erzeugte Abbildung dem Programmiersystem für die Programmierung der SPS bereitstellt.

Um als universelles Netzwerk für die Automatisierung dienen zu können, muss ein Ethernet-basierter Feldbus die beiden Welten abdecken und eine umfassende Einbindung aller Komponenten zur Konfiguration und Parametrierung sowohl offline als auch online erlauben.

3.2 Ziel der Arbeit

Ausgangssituation und Ziel der Arbeit sind abstrahiert in Bild 3.1 dargestellt. Ziel der Arbeit ist es, einen Beitrag dazu zu leisten, die Komplexität bei Inbetriebnahme und Rekonfiguration von Maschinen und Anlagen, die durch die hohe Flexibilität und den Variantenreichtum heutiger Automatisierungsgeräte und der durch sie bereit gestellten Funktionen entsteht, einfacher zu bewältigen.

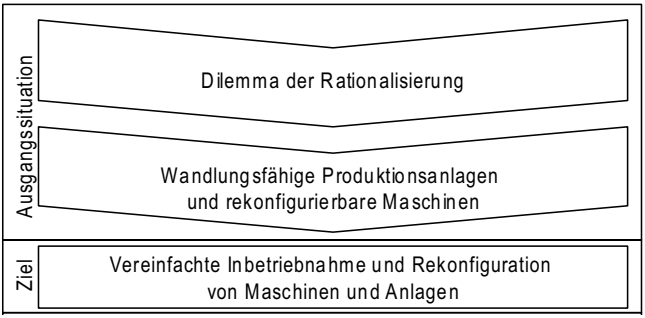


Bild 3.1: Ausgangssituation und Ziel der Arbeit

Um dieses Ziel zu erreichen, wird die in Bild 3.2 gezeigte Grundidee verfolgt.

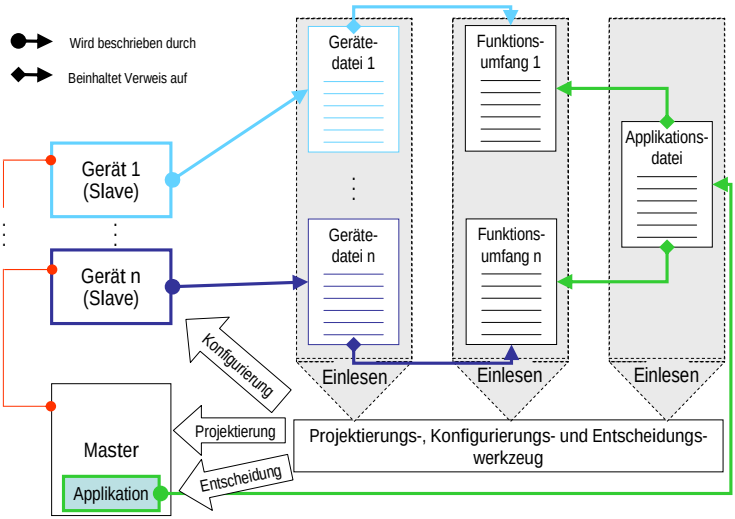


Bild 3.2: Motivation für die funktionale Typisierung von Automatisierungsgeräten

Die Geräte in einer Anlage können jeweils durch Gerätebeschreibungsdateien beschrieben werden. Diese beinhalten einen Verweis auf den Funktionsumfang, den das jeweilige Gerät zur Verfügung stellt.

Die gewünschte Applikation lässt sich ebenfalls in einer Beschreibungsdatei darstellen. Diese Beschreibungsdatei beinhaltet die Anforderungen der Applikation an durch bestimmte Geräte bereitzustellende Funktionen.

Werden alle diese Informationen einem Softwarewerkzeug bereitgestellt, so kann dieses prüfen, ob die durch die Applikation gestellten Anforderungen durch bestimmte Geräte erbracht werden können. Ebenfalls ist es damit möglich, eine Auswahl an Geräten automatisiert vorzuschlagen, die den Anforderungen genügen.

3.3 Anforderungen an ein Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten

Um das in 3.2 aufgezeigte Szenario erfüllen zu können, muss ein Informationsmodell existieren, das es erlaubt, Automatisierungsgeräte funktional zu beschreiben. Dieses muss folgende Anforderungen erfüllen:

- Es müssen sowohl homogene Automatisierungsgeräte, wie reine Antriebsgeräte oder reine E/A-Geräte, als auch hybride Automatisierungsgeräte, wie Antriebsgeräte mit integrierten Ein- und Ausgängen, dargestellt werden können.
- Die Funktionen, die ein Gerät für andere Teilnehmer des Netzwerkes bereit stellt, müssen darstellbar sein.
- Das Modell muss eine Typisierung der Geräte erlauben.
- Um eine Profilbildung zu erlauben, müssen die Funktionen in den Geräten in dem Modell klassifizierbar sein.
- Das Modell muss es erlauben, funktionale Zusammenhänge darstellen zu können.

3.4 Vorgehen in der Arbeit

Das Vorgehen in der Arbeit und Anwendung der Ergebnisse der Arbeit sind in Bild 3.3 dargestellt.

Zunächst werden der Stand der Technik und der Stand der Forschung auf dem Gebiet der Integrationsmöglichkeiten der notwenigen Informationen zur Darstellung von Automatisierungsgeräten im Engineeringprozess betrachtet. Außerdem werden die zum Verständnis der Arbeit notwenigen Grundlagen aus der Informatik und der Mathematik eingeführt.

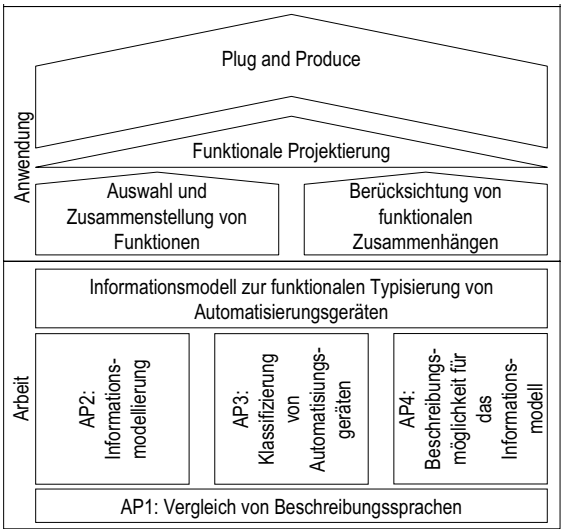


Bild 3.3: Aufbau der Arbeit und deren Anwendung

3.4.1 Methode zum Vergleich von Beschreibungssprachen

Um die unterschiedlichen Beschreibungssprachen und Mechanismen, die existieren, um Automatisierungsgeräte in ein Engineeringwerkzeug zu integrieren, vergleichen und bewerten zu können, wird eine Methode hergeleitet. Diese Methode soll es erlauben, das diesen Sprachen und Mechanismen zugrunde liegende Informationsmodell abzuleiten. Dies ist notwendig, da ein echter Vergleich der Sprachen nur auf der Ebene des zugrunde liegenden logischen Modells möglich ist. Die Formalisierung des Informationsmodells in Form einer Sprache und damit in Syntax und Semantik ist nur eine Form der Umsetzung, sagt aber nichts darüber aus, was mit der Sprache beschrieben werden kann.

Im Moment existiert keine systematische Methode, die es erlaubt, Beschreibungssprachen und ihre Mächtigkeit zu vergleichen. Ein solcher Vergleich ist beispielsweise dann notwendig, wenn die Beschreibung eines Automatisierungsgerätes von einer Beschreibungsform in eine andere überführt werden soll. Hierzu muss getestet

werden, ob es möglich ist, alle ursprünglich beschriebenen Eigenschaften auch in der neuen Darstellung zu beschreiben.

Ein weiterer Anwendungsbereich ist die Erweiterung eines Gerätemodells anhand von neuen Anforderungen. Mit der vorgestellten Methode kann getestet werden, ob das so entstandene Modell mit der bereits bestehenden Beschreibungssprache weiterhin beschreibbar ist.

Der dritte Anwendungsfall für eine solche Methode ist der Fall, in dem ein neues Modell entstanden ist. Es soll getestet werden, ob dieses mit bereits vorhandenen Beschreibungssprachen beschrieben werden kann.

Unter Zuhilfenahme dieser Methode werden in einem weiteren Schritt die vorhandenen Sprachen und Mechanismen verglichen und bezüglich ihrer Eignung zur Erreichung des in 3.2 definierten Ziels bewertet.

3.4.2 Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten

In einem weiteren Schritt wird ein Informationsmodell abgeleitet, das es erlaubt, Automatisierungsgeräte funktional zu typisieren. Das bedeutet, dass ein Informationsmodell entworfen wird, das in der Lage ist, die Funktionen, die ein Automatisierungsgerät anderen Teilnehmern im Netzwerk zur Verfügung stellt, abzubilden. Die Einteilung der Automatisierungsgeräte erfolgt dann nach ihren funktionalen Eigenschaften.

3.4.3 Funktionale Klassifikation von Automatisierungsgeräten

Zur Einteilung der Automatisierungsgeräte in funktional orientierte Typen wird ein Vorgehen hergeleitet. Dieses Vorgehen ermöglicht es, durch Automatisierungsgeräte in Form von Kommunikationsobjekten bereitgestellte Funktionen so in Funktionsmengen zu ordnen, dass sich die durch ein Gerät bereitgestellten Schnittstellen direkt durch die Kombination von Implementierungen dieser Funktionsmengen ergeben.

3.4.4 Beschreibungsmöglichkeit für das Informationsmodell

Abschließend wird dargelegt, wie aus den hergeleiteten Informationsmodellen eine Geräte- und eine Profilbeschreibungssprache entstehen können.

3.5 Zusammenfassung

Die Zusammenfassung des Kapitels „Anforderungen“ ist Bild 3.4 dargestellt. In diesem Kapitel wurde zunächst die Ausgangssituation beschrieben. Die Notwendigkeit, die hohe Produktivität der Großserienfertigung und die Flexibilität der Einzelteilfertig zu vereinigen, führt dazu, dass Maschinen und Anlagen heute zunehmend aus Baukästen konfiguriert werden. Es existiert die Forderung nach Kostenoptimierung durch Minimierung der notwenigen Zeiten für Inbetriebnahme und Rekonfiguration. Der dafür erforderliche Konfigurierungsvorgang wird heute durch Engineeringwerkzeuge unterstützt.

Ziel der Arbeit ist es, diesen Konfigurationsvorgang zu erleichtern, indem dem Engineeringwerkzeug über die heute zur Verfügung gestellten Informationen hinaus weitere bereitgestellt werden. Diese Informationen beinhalten funktionale Zusammenhänge und funktionale Eigenschaften der Automatisierungsgeräte.

Um dies zu erreichen, muss das Informationsmodell, auf welches das Engineeringwerkzeug zurückgreift, die Möglichkeit haben, alle Arten von Automatisierungsgeräten abbilden zu können. Um dieses Ziel zu erreichen, wird folgendes Vorgehen gewählt, welches den Arbeitspunkten dieser Arbeit entspricht:

1. Erarbeiten einer Methode zum Vergleich von allgemeinen Beschreibungssprachen.

2. Ableiten eines Informationsmodells zur funktionalen Typisierung von Automatisierungsgeräten.

3. Erstellung einer funktionalen Klassifikation von Automatisierungsgeräten.

4. Ableiten einer Beschreibungsmöglichkeit für das Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten.

Zur Erreichung des Ziels der Arbeit werden Methoden und Vorgehensweisen aus den Bereichen Mathematik, Informatik und Softwaretechnik angewandt. Das zentrale Paradigma in der Arbeit ist die Informationsmodellierung. Hinzu kommen die Aufgaben der Anforderungsanalyse, objektorientierter Entwurf und Mengentheorie.

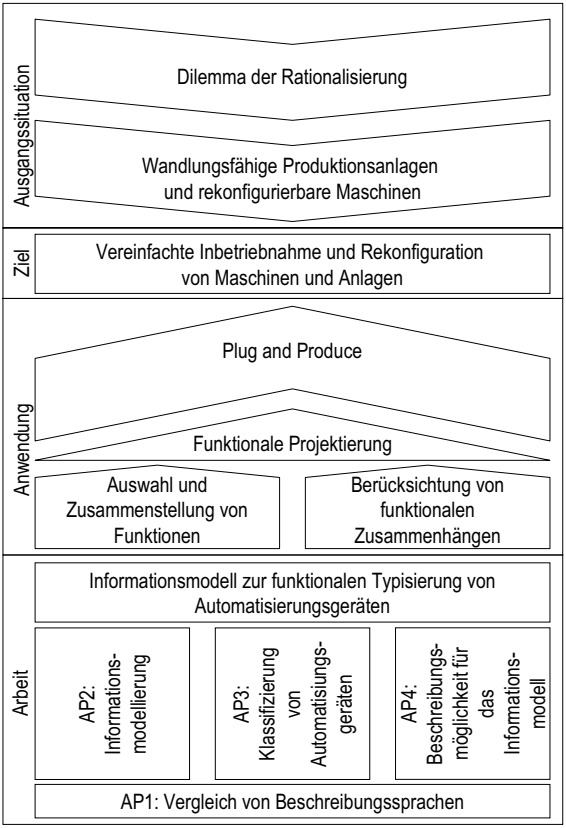


Bild 3.4: Zusammenfassung des Kapitels „Anforderungen“

4 Stand der Technik

In diesem Kapitel wird der Stand der Technik bezüglich der Integration von Geräteinformationen in ein Engineeringwerkzeug und die Spezifikation von funktionalen Schnittstellen von Automatisierungsgeräten vorgestellt. Hierzu werden zunächst die gängigen Formate von Gerätebeschreibungsdateien eingeführt. Danach werden unterschiedliche Gerätebeschreibungssprachen und andere hersteller-übergreifende Mechanismen zur Integration von Geräten in ein Engineeringwerkzeug betrachtet.

In weiteren Schritten werden Methoden zur formalen Spezifikation von Schnittstellen und die Graphentheorie eingeführt.

4.1 Gängige Dateiformate von Gerätebeschreibungsdateien

Zwei unterschiedliche Dateiformate sind für die Betrachtungen innerhalb der vorgelegten Arbeit relevant, das Initialisierungsdateiformat und das XML-Dateiformat.

4.1.1 Initialisierungsdateiformat

Eine Initialisierungsdatei besitzt ein strukturiertes Textformat und wird dazu benutzt, um Informationen wie beispielsweise Programmeinstellungen abzulegen /58/. Bild 4.1 zeigt den grundsätzlichen Aufbau einer Initialisierungsdatei.

Es gibt die Möglichkeit, die Datei in unterschiedlich bezeichnete Sektionen zu unterteilen, in dem der Name in eckigen Klammern der Sektion vorangestellt wird. Informationen werden als Variablen mit entsprechenden Werten dargestellt. Variablennamen müssen eindeutig innerhalb einer Sektion sein. Kommentare werden durch das Voranstellen eines Semikolons gekennzeichnet.

```
;Freier Kommentar

[Sektion_1]
;Freier Kommentar
Schlüssel_1=Wert_1
Schlüssel_2=Wert_2

...

[Sektion_2]
;Kommentar
Schlüssel_1=Wert_3
```

Bild 4.1: Aufbau einer Initialisierungsdatei

4.1.2 XML-Dateiformat

Extensible Markup Language (XML) /59/ bedeutet auf Deutsch erweiterbare Beschreibungssprache. Sie wird dazu verwendet, Formate zum Speichern von beliebigen strukturierten Informationen zu schaffen. Das zugrunde liegende Datenmodell wird in einer Schemadefinition beschrieben. Gängige Sprachen zur Beschreibung von Schemata sind XML-Schema /60/, Relax NG /61/ und Schematron /62/.

Eine XML-Datei ist hierarchisch strukturiert und bildet logisch eine Baumstruktur. Die Elemente werden mittels Start- und Endtag oder einem Empty-Elementtag ausgezeichnet. Jedem Start- bzw. Empty-Elementtag können zusätzlich Attribute zugeordnet werden.

Bei der Verarbeitung bietet XML große Vorteile gegenüber anderen Formaten wie beispielsweise dem Initialisierungsdateiformat, da für das Einlesen eine Vielzahl an verschiedenen Parsern und Prozessoren in allen erdenklichen Lizenzen zur Verfügung stehen /63/.

4.2 Gerätebeschreibungssprachen

In der Automatisierungstechnik existieren viele verschiedene Beschreibungssprachen, welche unterschiedliche Aufgaben innerhalb des Lebenszyklus der Automatisierungsanlage und der Automatisierungspyramide wahrnehmen /65/. Im Folgenden werden die

wichtigsten derzeit existierenden Beschreibungssprachen für die Automatisierungstechnik betrachtet.

4.2.1 Generic Station Description und Generic Station Description Markup Language

Die Gerätebeschreibungssprachen GSD /66/ und GSDML /67/ werden von der Profibus Nutzer Organisation e.V. (PNO) spezifiziert. Mit GSD-Dateien werden Automatisierungsgeräte beschrieben, die eine Profibusschnittstelle besitzen, mit GSDML-Dateien können zusätzlich auch noch Automatisierungsgeräte beschrieben werden, welche eine Profinetschnittstelle besitzen.

GSD nutzt als Dateiformat das Format der Initialisierungsdateien. GSDML setzt als Weiterentwicklung von GSD auf XML auf /68/.

4.2.2 Field Device Configuration Markup Language

Die Gerätebeschreibungssprache FDCML wurde als systemneutrale Beschreibungssprache für Automatisierungskomponenten entwickelt. Sie wird heute hauptsächlich dazu verwendet, um Interbus- und Drivecomkomponenten zu beschreiben.

FDCML verwendet XML als Dateiformat und ist bei der IEC standardisiert /50/.

4.2.3 Electronic Data Sheet

EDS wurde ursprünglich dazu verwendet, um Automatisierungsgeräte zu beschreiben, die eine CAN-basierte Kommunikationsanschaltung besitzen. Inzwischen hat sich der Anwendungsbereich dahingehend erweitert, dass Geräte beschrieben werden, die ein CANopen Profil besitzen, wie beispielsweise CoE (CANopen over Ethercat) oder POWERLINK-Geräte /69/.

4.3 Herstellerübergreifende Mechanismen zur Integration von Geräten in ein Engineeringwerkzeug

Neben den Gerätebeschreibungssprachen existieren Mechanismen, die dazu dienen, den Zugriff auf Geräte zur Parametrierung und Konfigurierung von Geräten durch ein Engineeringwerkzeug zu ermöglichen.

Diese werden oft als Gerätebeschreibungssprachen bezeichnet, sind aber im engeren Sinn keine Gerätebeschreibungssprachen, sondern beschreiben nur, wie ein Engineeringwerkzeug auf diese Geräte zugreifen kann und nicht wie sich diese im Betrieb verhalten.

4.3.1 Electronic Device Description Language

Heutige Automatisierungsgeräte besitzen sehr viele unterschiedliche Einstellmöglichkeiten, um sie für einen individuellen Einsatzfall zu parametrieren. Um diese Parametrierung bei der Inbetriebnahme und eine Überwachung im laufenden Betrieb durch einen Nutzer zu ermöglichen, bieten sie typischerweise eine digitale Kommunikationsschnittstelle. Die Vorgänge werden meist durch ein hersteller- oder sogar produktspezifisches Werkzeug unterschützt. Um nicht für jeden Gerätetyp ein eigenes Werkzeug erstellen zu müssen, wurde Anfangs der 90er Jahre des 20. Jahrhunderts DDL entwickelt /70/.

Ziel war ein einzelnes Softwarewerkzeug, welches durch Interpretation einer Datei viele unterschiedliche Gerätetypen bedienen und parametrieren kann. Der grundlegende Gedanke ist, dass das Erstellen der Beschreibung mittels einer Beschreibungsdatei weniger aufwändig ist als das Erstellen eines gerätetypspezifischen Softwarewerkzeuges. Unterschiedliche Ansätze in dieser Richtung wurden zusammengelegt. Das Ergebnis dieser Konsolidierung ist unter der Bezeichnung Electronic Device Description Language (EDDL), ein IEC Standard /71/.

Der Fokus von EDDL liegt in der Parametrierung und der Visualisierung der Geräte im laufenden Betrieb, lässt aber die Aspekte der Konfiguration und der durch das Gerät bereitgestellten Funktionen außer Acht.

4.3.2 Field Device Tools/Device Type Manager

Moderne Feldgeräte bieten die Möglichkeit, mittels Computerprogrammen, die eigens für das jeweilige Feldgerät entwickelt wurden, Parametrierung und Diagnose zu betreiben. Um aber nicht für jedes Feldgerät ein eigenes Programm auf einem Diagnoserechner installieren zu müssen, wurde das herstellerübergreifende FDT/DTM-Konzept entwickelt /72/, /73/.

Ein Device Type Manager (DTM) ist ein Treiber für ein Feldgerät /74/. Der DTM umfasst alle Funktionen, die Struktur, die Parametrierung sowie die GUI inklusive einem Hilfesystem für ein bestimmtes Feldgerät oder evtl. eine Gerätefamilie. Er wird als Programm auf dem PC installiert, kann aber nur aus einem Container, auch Rahmenapplikation (Frame Application) genannt, gestartet werden.

Der Begriff Field Device Tool (FDT) steht für eine Schnittstellendefinition /74/. Damit alle DTMs von verschiedenen Herstellern in jedem Container korrekt funktionieren, müssen die Schnittstellen zum umgebenden System sowie zu anderen DTMs klar definiert werden. Diese Schnittstellendefinition wird FDT genannt.

Das erste aller Rahmenprogramme war PACTware (Process Automation Configuration Tool). PACTware wurde gemeinsam von diversen Feldgeräteherstellern und Systemlieferanten entwickelt /75/. Mittlerweile gibt es auch andere FDT-Frames, die den Bedürfnissen der verschiedenen Anwender gerecht werden, zum Beispiel FieldCare von Endress+Hauser /76/, fdtCONTAINER von M&M /77/, SMART VISION von ABB /78/ und SYCON.net der Firma Hilscher /79/.

Das FDT/DTM-Konzept dient dazu, um hersteller- oder sogar produktspezifische Software zur Parametrierung und Visualisierung von Feldgeräten (DTM) in ein Engineeringwerkzeug eines anderen Herstellers (FDT) zu integrieren. Dazu wird durch das Engineeringwerkzeug ein Rahmen in Form von definierten Schnittstellen zur Verfügung gestellt. Die Software, die sich in das Rahmenprogramm integrieren möchte, benutzt diese Schnittstellen, unter anderem auch, um mit dem Feldgerät zu kommunizieren. Das FDT/DTM-Konzept sieht eine geräteübergreifende Konfiguration und die Definition von Geräteschnittstellen nicht vor.

4.4 Formale Spezifikation von Schnittstellen

Eine formale Spezifikation ist eine in einer formalen Notation geschriebene Spezifikation. Die der Beschreibung zugrunde liegenden mathematischen Konzepte werden dazu benutzt, um die Syntax und Semantik der beschreibenden Notation zu definieren.

Die formale Spezifikation beschreibt eine abstrakte Vorstellung eines Teils eines Systems. Das so Beschriebene kann meist auf mehrere Arten implementiert werden, da nur das Verhalten bezüglich der Schnittstellen nach außen beschrieben wird.

Durch immer weitere Verfeinerung der Beschreibung kann die Menge der formal richtigen Umsetzungen weiter eingeschränkt werden. Dies kann beispielsweise dadurch geschehen, dass nicht nur die Schnittstellen nach außen beschrieben werden, sondern auch das System und dessen interne Zustände selbst.

Aus einer formalen Spezifikation können stets Prüfregeln abgeleitet werden, um eine Umsetzung gegenüber der formalen Spezifikation testen zu können /80/, /81/. Die Art der Beschreibungsform kann unterschiedliche Ebenen der Verifikation der Umsetzung gegenüber der Spezifikation ermöglichen:

- Bei der **Programmverifikation** wird mit den Mitteln der formalen Logik nachgewiesen, dass eine Implementierung die Verfeinerung einer formalen Spezifikation ist und alle Anforderungen erfüllt, die diese übergeordnete Spezifikation erfüllt /82/.
- Bei der **Designverifikation** wird mit den Mitteln der formalen Logik nachgewiesen, dass eine Feinspezifikation die Verfeinerung einer Ausgangsspezifikation ist und alle Anforderungen an die Ausgangsspezifikation ebenfalls erfüllt /82/.

Die wichtigsten Klassen formaler Spezifikationen sind in den folgenden Unterkapiteln beschrieben. Mit allen beschriebenen Klassen können sowohl eine Programmverifikation als auch eine Designverifikation durchgeführt werden.

4.4.1 Mathematische Spezifikation

Eine mathematische Spezifikation beschreibt das Verhalten eines Systems und dessen Zustände und gültigen Zustandsübergänge durch eine Berechnungsvorschrift auf niedrigem Abstraktionsniveau /82/. Hierbei kommen sogenannte Vor- und Nachbedingungen zum Einsatz. Eine Nachbedingung ist ein Zustand, welcher für das beschriebene System erlaubt ist, nachdem eine bestimmte Aktion ausgeführt wurde. Vorbedingungen beschreiben die Zustände, die zu einer bestimmten Aktion führen können.

Typische Einsatzgebiete für die mathematische Spezifikation ist die Entwurfsphase von abgeschlossenen kleinen Softwareteilen, wie kleinen Programmen oder einzelnen Prozeduren.

4.4.2 Axiomatische Spezifikation

Die Methode der axiomatischen Spezifikation definiert die Semantik von Funktionen und Objekten durch die Beschreibung ihrer Beziehungen untereinander /82/. Dies geschieht in prädikatenlogischer Form. Es ist möglich, mittels axiomatischer Spezifikation Funktionen und Objekte auf jeder Abstraktionsebene zu beschreiben.

Haupteinsatzgebiete der axiomatischen Spezifikation sind die Spezifikation von Sicherheitsmodellen und die Beschreibung abstrakter Datentypen und ihrer Abhängigkeiten.

4.4.3 Algorithmische Spezifikation

Die algorithmische Spezifikation beschreibt das Verhalten eines Systems oder Systemteils durch eine oder mehrere genaue Berechnungsvorschriften /82/. Dadurch ist sie meist auf einem niedrigen Abstraktionsniveau der Spezifikationsarbeit zu finden. Der Vorteil dieser Art von Spezifikation ist die direkte Umsetzbarkeit der Spezifikation in Programmcode durch das meist bereits sehr niedrige Abstraktionsniveau.

Diese Art der Spezifikation wird meist kurz vor der Umsetzung einer Spezifikation im Programmcode verwendet. Dann dient die algorithmische Spezifikation direkt als Programmiervorgabe.

4.4.4 Spezifikation mittels Traces

Die Spezifikation mittels Traces stellt die zeitliche Reihenfolge von Zuständen oder Operationen und ihre Eigenschaften dar /82/. Anwendbar ist diese Art der Beschreibung auf jeder Abstraktionsebene eines Systems.

Sie wird hauptsächlich für die für die Beschreibung nebenläufiger Prozesse oder komplexer Automaten verwendet.

4.4.5 Spezifikation mittels temporaler Logik

Bei der Spezifikation mittels temporaler Logik werden diskreten Zeitpunkten Zustände zugeordnet /82/. Die temporale Logik besteht aus Aussagen, die abhängig vom Zeitpunkt richtig oder falsch sind. Mittels temporaler Logik kann somit abhängig vom Zeitpunkt eine Aussage über die Zustände des Systems gemacht werden, als auch über die Abfolge der Zustände im System.

Diese Art der Beschreibung ist geeignet für Aussagen über die Reihenfolge bei Abläufen innerhalb von Programmen oder Systemen.

4.5 Graphentheorie

Die **Graphentheorie** ist ein Teilgebiet der Mathematik. Ziel dieser Theorie ist die Untersuchung der Eigenschaften von Graphen /83/. In der Graphentheorie ist ein Graph $G = (V, E)$ durch eine Menge von Knoten V, die eine Menge von Kanten E miteinander verbunden sind, definiert.

Die Graphentheorie klassifiziert Graphen grundsätzlich nach folgenden Eigenschaften: Der Wertigkeit, der Komplexität und der Eigenschaft, ob ein Graph gerichtet ist.

- **Endliche Graphen** haben eine endliche Anzahl an Knoten und Kanten. Für **unendlichen Graphen** trifft diese Eigenschaft nicht zu, sie besitzen eine unendliche Anzahl an Knoten oder Kanten.
- Bei **gerichteten Graphen** sind die Kanten gerichtet, was durch Pfeile statt Linien als Kanten dargestellt wird. **Ungerichtete Graphen** besitzen nur Linien als Kanten.
- Bei sogenannten **einfachen Graphen** existiert zur Verbindung zweier Knoten höchstens eine Kante. Außerdem sind an einer Kante immer höchstens zwei Knoten beteiligt. Bei **komplexen Graphen** bestehen diese Einschränkungen nicht. Komplexe Graphen, bei denen Knoten durch mehr als eine Kante verbunden sein können, werden **Multigraphen** genannt. Komplexe Graphen, die mittels einer Kante mehr als zwei Knoten verbinden können, werden **Hypergraphen** genannt.

4.6 Zusammenfassung und Bewertung

In diesem Kapitel wurde der Stand der Technik beleuchtet. Hierzu wurden zunächst die heute gängigen Dateiformate von Konfigurationsdateien, das Initialisierungsdateiformat und das XML-Dateiformat betrachtet.

Im Weiteren wurden heute verbreitete Gerätebeschreibungssprachen und hersteller-übergreifende Mechanismen zur Integration von Geräten in Engineeringwerkzeugen vorgestellt. Hierbei wurden die folgenden Sprachen und Mechanismen betrachtet:

- Generic Station Description
- Generic Station Description Markup Language
- Field Device Configuration Markup Language
- Electronic Data Sheet
- Electronic Device Description Language
- Field Device Tools/Device Type Manager

Danach wurden Mechanismen zur Spezifikation von Schnittstellen und die Graphentheorie eingeführt. Diese Mechanismen werden in den folgenden Kapiteln zur Herleitung der Ergebnisse verwendet.

In der Automatisierungstechnik existiert eine Vielzahl von unterschiedlichen Gerätebeschreibungssprachen. Allen ist gemein, dass sie ursprünglich für einen Spezialfall entworfen worden sind. Durch die Konvergenz der dedizierten Feld- und Antriebsbusse zu universellen Netzwerken für die Automatisierung werden die Anforderungen an Gerätebeschreibungssprachen vielschichtiger. Die zugrunde liegenden Informationsmodelle müssen hierfür gegebenenfalls angepasst werden, da diese auf spezielle Anwendungsfälle zugeschnitten worden sind. Diese Untersuchung wurde bisher noch nicht durchgeführt.

Um bewerten zu können, ob die Gerätebeschreibungssprachen den in 3.3 abgeleiteten Anforderungen nach der Darstellbarkeit von reinen Antriebs- und E/A-Geräten wie auch Hybridgeräte und der Beschreibbarkeit von Funktionen des Gerätes gerecht werden, existiert heute keine systematische Methode. Eine solche Methode wird als erster Arbeitspunkt im nächsten Kapitel abgeleitet.

5 Methode zum Vergleich von allgemeinen Beschreibungssprachen

In der Automatisierungstechnik existieren vielfältige unterschiedliche Beschreibungssprachen. Im Moment existiert keine systematische Methode, die es erlaubt, Beschreibungssprachen und ihre Mächtigkeit zu vergleichen.

Um eine solche Methode abzuleiten, werden Methoden aus der Informationsmodellierung angewendet. Außerdem werden mengentheoretische Beziehungen aus Mathematik und Informatik betrachtet und an den speziellen Anwendungsfall angepasst.

5.1 Mächtigkeit einer Gerätebeschreibungssprache

Die Mächtigkeit einer Gerätebeschreibungssprache zeichnet sich dadurch aus, was mit dieser Gerätebeschreibungssprache beschrieben werden kann. In der Mathematik und der Informatik bezeichnet der Begriff Kardialität oder Mächtigkeit einer Menge ein Maß für die Größe der Menge /84/. Für den Vergleich der Mächtigkeit von Menge existieren folgende Definitionen:

1. $|A| \leq |B|$, falls eine injektive totale Abbildung $f: A \rightarrow B$ existiert.

2. $|A| < |B|$, falls keine surjektive totale Abbildung $f: A \rightarrow B$ existiert.

3. $|A| = |B|$, falls eine bijektive totale Abbildung $f: A \rightarrow B$ existiert.

Durch die Übertragung dieser Definition aus Mathematik und Informatik lassen sich Gerätebeschreibungssprachen entsprechend der Mengenlehre vergleichen:

- Zwei Gerätebeschreibungssprachen GS_1 und GS_2 sind dann gleichmächtig, wenn eine Abbildung $f: GS_1 \rightarrow GS_2$ existiert, die umkehrbar ist.
- Eine Gerätebeschreibungssprache GS_1 ist dann mächtiger als GS_2, wenn es eine Abbildung eines Teils der beschreibbaren Elemente aus GS_1 nach GS_2 gibt, nicht aber umgekehrt.

5.2 Die Entstehung einer Gerätebeschreibungssprache

Bei der Entstehung einer Gerätebeschreibungssprache wird so verfahren, dass zunächst die Summe aller zu beschreibenden Geräte zu einem abstrakten Gerät verallgemeinert wird. Dieses abstrakte Gerät bildet dann die Grundlage des Gerätemodells. Das Gerätemodell wird durch eine Beschreibungssprache in eine formalisierte Form gebracht.

Eine Gerätebeschreibung beschreibt ein konkretes Gerät oder eine konkrete Menge von Geräten, die aus Sicht des Engineeringwerkzeugs gleich sind. Eine Gerätebeschreibungssprache muss in der Lage sein, eine Menge von Geräten mit unterschiedlichen Eigenschaften aus Sicht des Engineeringwerkzeugs darzustellen. Daher werden im Definitionsprozess die abzubildenden Geräte zu einer Menge abstrahiert. Für diese Menge an Geräten wird dann eine Beschreibungssprache entworfen, die es erlaubt, alle konkreten Geräte zu beschreiben. Zur Definition der Beschreibungssprache wird ein logisches Modell, ein Gerätemodell, benötigt. Dies entsteht durch Abstraktion zu einem mentalen Modell. Dieses mentale Modell entspricht der Sicht des Engineeringwerkzeugs auf die Geräte und bildet somit nur einen relevanten Teil der Wirklichkeit ab. Somit handelt es sich bei diesem Vorgehen um eine Informationsmodellierung nach der Definition in Kapitel 2.7.

Oftmals wird dieser Abbildungsprozess durch Methoden aus der Objektorientierung unterstützt. Dabei werden die realen Geräte als Objekte und deren Eigenschaften abgebildet.

Dieses Vorgehen erstreckt sich auf drei Ebenen: der Einzelgeräteebene, der Metaebene und der Modellebene wie in Bild 5.1 gezeigt wird.

- Auf der **Einzelgerätebene** wird die Information festgelegt, die das einzelne Gerät beschreibt.
- Auf der **Metaebene** wird die Informationsmenge einer abstrakten Gerätemenge transportiert, um die Gerätebeschreibungssprache zu definieren.
- Auf **Modellebene** wird das Mentale Modell durch Formalisierung in ein Logisches Modell – das Gerätemodell – transformiert.

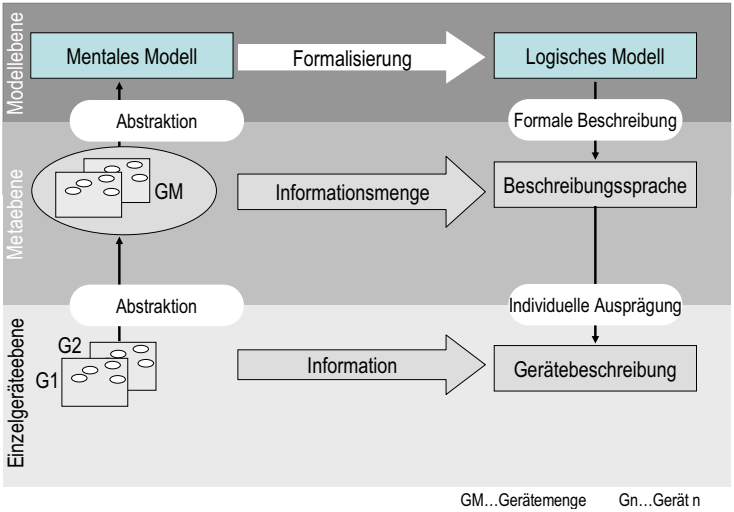


Bild 5.1: Vom Gerät zur Gerätebeschreibung

Der Übergang zwischen diesen Ebenen erfolgt nach oben immer durch Abstraktion. Das heißt im konkreten Fall des Übergangs zwischen der Einzelgeräte- und der Metaebene durch Abstraktion einer Menge von Einzelgeräten zu einer abstrakten Gerätemenge. Der Übergang von der Metaebene auf die Modellebene erfolgt durch die Abstraktion der abstrakten Gerätemengen zum Mentalen Modell, welches der individuellen Vorstellung von einem verallgemeinerten Gerät der betrachteten Menge an Geräten entspricht.

Der Übergang in die andere Richtung erfolgt jeweils durch Konkretisierung. So wird der Übergang zwischen der Modellebene und der Metaebene durch Konkretisierung in Form der Formalisierung des Logischen Modells durch die Zuweisung von Syntax und Semantik zur Gerätebeschreibungssprache vollzogen. Der Übergang von der Metaebene zur Gerätebene erfolgt durch Konkretisierung in Form von individueller Ausprägung, das heißt der Beschreibung des Gerätes selbst. Diese Schritte sind allerdings nicht jedes Mal notwendig, wenn ein Gerät durch eine Gerätebeschreibungsdatei beschrieben werden soll. Durch den Abstraktionsschritt von der Einzelgeräte- auf die Metaebene von einzelnen Geräten zur Menge aller betrachteten Geräte enthält die Gerätebeschreibungssprache die gesamte zur Beschreibung notwendige Informationsmenge.

Ist der Weg zur Gerätebeschreibungssprache und somit sowohl die Gerätebeschreibungssprache und das zugrunde liegende Logische Modell – das Gerätemodell – einmal festgelegt, so lässt sich jedes Gerät, das in das Logische Modell hineinpasst, allein durch die Konkretisierung in Form von individueller Ausprägung beschreiben, da die Teilergebnisse der anderen Strecken des Weges dadurch nicht verändert werden.

5.3 Ableiten der Methode zum Vergleich allgemeiner Beschreibungssprachen

5.3.1 Ausgangssituation

Die Mächtigkeit einer Beschreibungssprache wird durch das zugrunde liegende Gerätemodell und die Informationsmenge, die es transportieren soll, festgelegt. Das Problem hierbei ist, dass nur die Spezifikationen der Sprachen und Beschreibungen von einzelnen Geräten vorliegen, nicht aber die logischen Modelle. Sie sind in der Spezifikation weder formal noch inhaltlich beschrieben. Daher lassen sich die einzelnen Beschreibungssprachen nicht auf der Spezifikation methodisch vergleichen.

Ein wirklicher Vergleich der Sprachen kann nur auf logischer Modellebene passieren, da die Syntax der Sprache und ihre Semantik nur eine Frage der Umsetzung sind. Deswegen muss zum Ableiten der Mächtigkeit und des Umfangs von Gerätebeschreibungssprachen ein Weg hin zum Logischen Modell gefunden und gegangen werden.

5.3.2 Bewertung von Methoden zum Vergleich von Beschreibungssprachen

Der ursprüngliche Weg vom Gerät auf der Einzelgeräteebene über die Metaebene zum Mentalen Modell, aus dem das Logische Modell hervorgeht, ist für bereits bestehende Gerätebeschreibungssprachen im Nachhinein nicht mehr nachvollziehbar. Die meisten Schritte, die auf dem Weg gegangen wurden, sind nur mental abgelaufen und nicht dokumentiert. Somit muss ein anderer Weg gefunden werden, um das Logische Modell herzuleiten.

Zum Ableiten des Logischen Modells stehen folgende mögliche Ausgangspunkte zur Verfügung:

- die Syntax und die Semantik der zu betrachtenden Gerätebeschreibungssprache,
- die vorhandenen Geräte und
- die konkrete Beschreibungen vorhandener Geräte.

Der am nächsten liegende vorhandene Ausgangspunkt ist die Gerätebeschreibungssprache selbst. Von diesem Punkt aus muss die formale Beschreibung umgekehrt werden, um zum Logischen Modell zu gelangen. Wird als Ausgangspunkt die Gerätebeschreibungssprache gewählt, so stellt sich das Problem, dass, wie bereits erwähnt, die Syntax und die Semantik der einzelnen Beschreibungssprache nur eine Frage der Umsetzung sind. Es ist also möglich, dass einzelne Sprachelemente innerhalb zweier Sprachen genau gleich aussehen, aber eine komplett andere Bedeutung haben. Andererseits ist es ebenso möglich, dass mit zwei anders aussehenden Sprachelementen dasselbe gemeint wird. Somit ist dieses Vorgehen sehr fehleranfällig und scheidet aus.

Eine andere Möglichkeit ist es, genau den Weg zu beschreiten, der ursprünglich bei der Entstehung der Beschreibungssprache auch gegangen wurde. Das heißt, die vorhandenen Geräte zunächst zu Gerätemengen zu abstrahieren, um dann durch weitere Abstraktion zum Mentalen Modell zu gelangen. Durch Formalisierung entsteht dann das gesuchte Logische Modell.

Dieses Vorgehen hat den Nachteil, dass es nicht garantiert, dass dasselbe Mentale Modell und somit auch dasselbe Logische Modell entsteht, wie das, welches der zu betrachtenden Beschreibungssprache zugrunde liegt, da die Schritte vom Gerät zum Logischen Modell alle sehr subjektiv sind. Um das Ergebnis nachzuprüfen, muss deswegen versucht werden, ob das abgeleitete Logische Modell durch Formalisierung in die zu betrachtende Gerätebeschreibungssprache überführt werden kann. Dieser Prozess muss dann iterativ so lange wiederholt werden, bis eine hinreichend gute Näherung erreicht ist. Da dieses Vorgehen sehr lange dauert und auf Grund der vielen Ungenauigkeiten nicht garantieren kann, dass am Ende das gesuchte Logische Modell gefunden wird, scheidet auch dieses aus.

Die dritte und letzte Möglichkeit ist, bei der konkreten Beschreibung einer Menge von Einzelgeräten anzusetzen und von dieser das Logische Modell abzuleiten. Das Vorgehen zum Ableiten der Mächtigkeit und des Umfangs von Gerätebeschreibungssprachen setzt an der konkreten Beschreibung von Einzelgeräten an und abstrahiert verschiedene individuelle Ausprägungen. Die Summe der Ergebnisse bildet dann die transportierte

Informationsmenge der betrachteten Gerätebeschreibungssprache. Damit lässt sich durch weitere Abstraktion das der Gerätebeschreibungssprache zugrunde liegende Logische Modell – das Gerätemodell – ableiten.

Auch dieses muss wieder durch Formalisierung in die Gerätebeschreibungssprache überführt werden und die Ergebnisse müssen iterativ abgeglichen werden. Dieses Vorgehen wird empfohlen, da es am wenigsten fehleranfällig ist und den besten und einfachsten Weg zum Ziel darstellt.

5.4 Vergleich und Bewertung heutiger Sprachen

Mittels der in Kapitel 5.3 abgeleiteter Methode werden die Gerätemodelle, die den Gerätebeschreibungssprachen Generic Station Description Generic (GSD), Station Description Markup Language (GSDML), Field Device Configuration Markup Language (FDCML) und Electronic Data Sheet (EDS) zu Grunde liegen, zunächst hergeleitet. Sind die Modelle vorhanden, so werden sie im Weiteren verglichen und dann bewertet.

5.4.1 Generic Station Description und Generic Station Description Markup Language

Die Gerätebeschreibungssprachen GSD und GSDML dienen dazu, Profibus- und Profinetgeräte für den Engineeringprozess zu beschreiben /85/. Profibusgeräte werden mittels GSD beschrieben. Der logische Aufbau eines Profibusgerätes ist in Bild 5.2 dargestellt. Es ist eine Sicht, die auf dem Aufbau von modularen E/A-Geräten basiert. Ein Gerät besteht aus einem Grundgerät, typischerweise einem Buskoppler und mehreren Modulen, die in sogenannte Slots eingesteckt werden können. Diese Slots besitzen sowohl zyklische als auch azyklische Daten und Dienste. Über das Grundgerät lassen sich Diagnosedaten abrufen, es hat normalerweise keine zyklischen Daten.

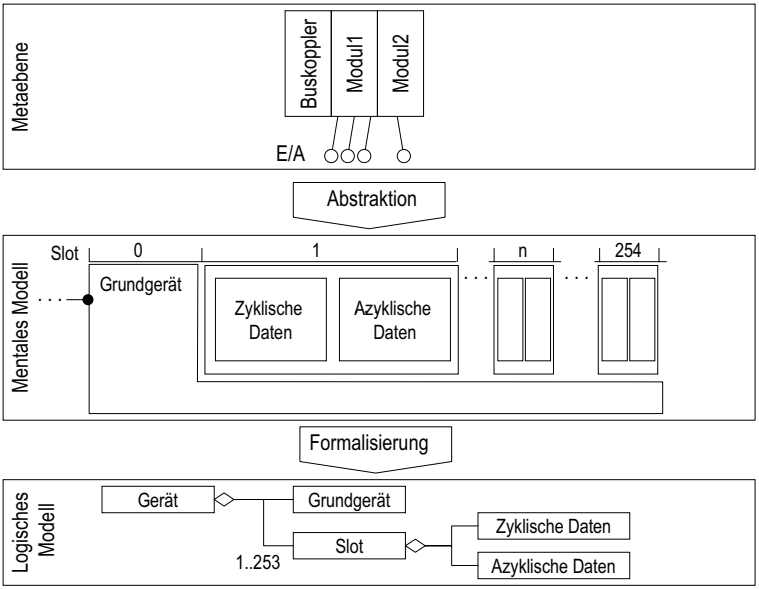


Bild 5.2: Logisches Modell eines Profibusgerätes

GSDML ist eine Weiterentwicklung der GSD. Dies zeigt sich auch im logischen Modell der Profinetgeräte. Auch ein Profinetgerät besteht ebenfalls aus Slots. Die Sicht bei Profinetgeräten wurde dahingehend erweitert, dass in einem Slot mehrere sogenannte Subslots existieren, die jeweils zyklische oder azyklische Daten und Dienste zur Verfügung stellen, wie es in Bild 5.3 dargestellt ist.

Das logische Modell, welches hinter GSD und GSDML steckt, bietet ein auf modularen E/A-Geräten beruhendes Modell. Es ist darin aber nicht vorgesehen, Profile oder Funktionsgruppen zu beschreiben.

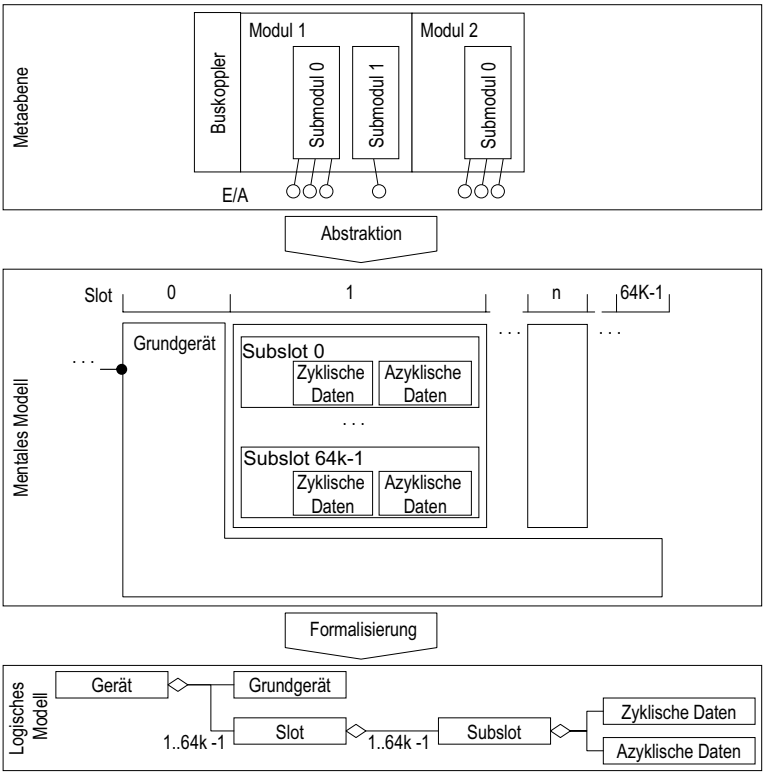


Bild 5.3: Logisches Modell eines Profinetgerätes

5.4.2 Field Device Configuration Markup Language

Der FDCML liegt ein generisches Gerätemodell zugrunde. Ein Gerät in diesem Modell besitzt Kommunikationszugriffspunkte, die sogenannten MAUs (Media Access Units). Diese MAUs sind jeweils über eine Kommunikationseigenschaft mit E/A-Daten und Geräteparameter verbunden. Die E/A-Daten lassen sich in Kanäle einbinden, wie es in Bild 5.4 gezeigt ist.

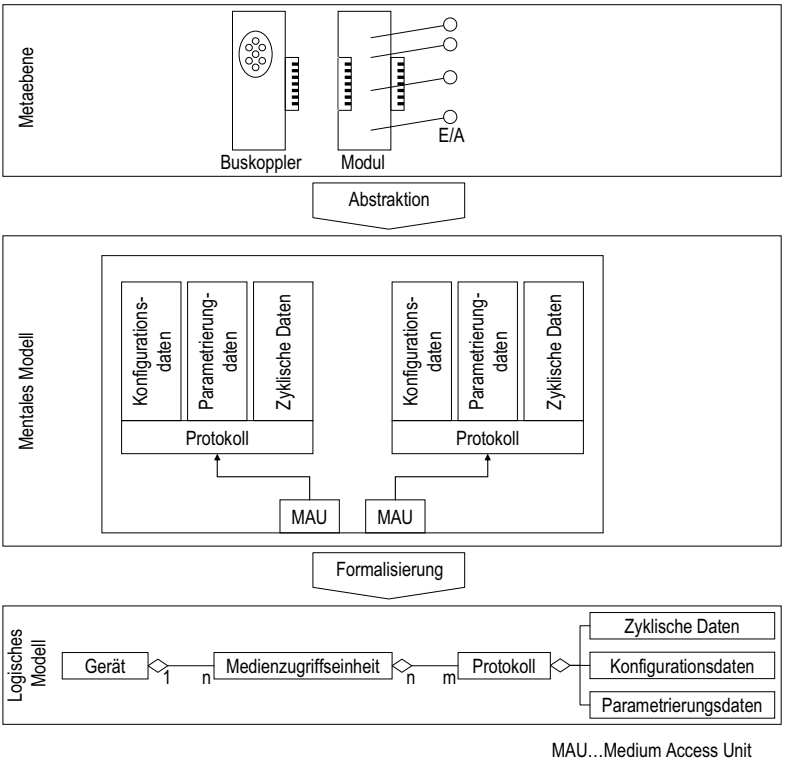


Bild 5.4: Logisches Modell nach FDCML

FDCML bietet eine sehr generische Sicht auf Geräte. Das Modell sieht sogar vor, Profile und deren Beschreibungen einzubinden, lässt aber offen, wie dies zu tun ist.

5.4.3 Electronic Data Sheet

Hinter EDS steckt kein einheitliches Informationsmodell. Es ist ausschließlich eine Sprache mit mehreren Dialekten, wie es in Bild 5.5 gezeigt ist.

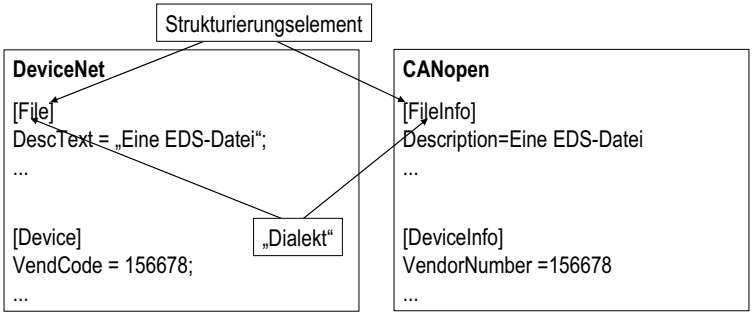


Bild 5.5: Unterschiedliche Dialekte von EDS

Der Aufbau einer EDS-Datei ist exemplarisch für DeviceNet-Geräte in Bild 5.6 dargestellt. Eine EDS-Datei besteht aus zwei Teilen, einem generell spezifizierten Inhalt, der Informationen über die Datei und das Gerät beinhaltet und einem unspezifizierten profilabhängigen Teil, in dem frei Informationen abgebildet werden können.

EDS bietet die Möglichkeit, sowohl das Gerät für den Engineeringprozess bereitzustellen als auch das konfigurierte Gerät zu beschreiben, eine Beschreibung von funktionalen Zusammenhängen ist aber nicht möglich.

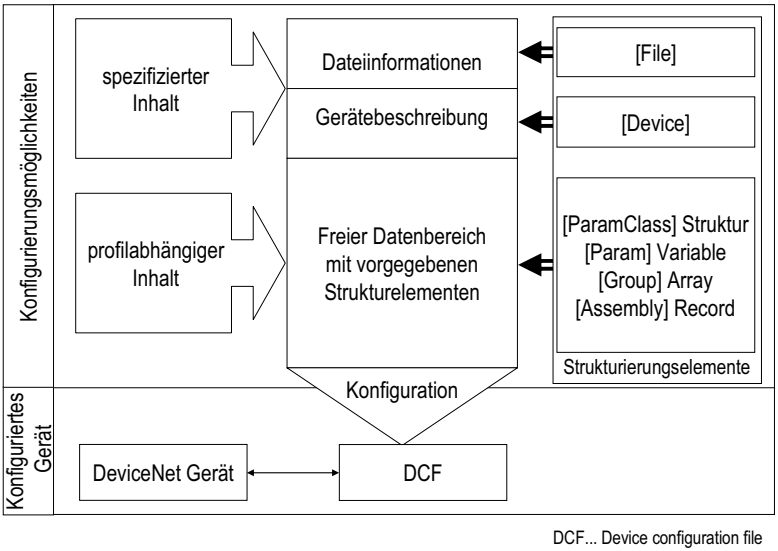


Bild 5.6: Aufbau einer Gerätebeschreibung für ein DeviceNet-Gerät

5.5 Zusammenfassung

In diesem Kapitel wurde eine Methode begründet, die es erlaubt, Beschreibungssprachen und ihre Mächtigkeit zu vergleichen. Dazu wurde zunächst der Begriff der Mächtigkeit einer Beschreibungssprache definiert. Der Begriff wurde aus der Mengentheorie entlehnt und den besonderen Randbedingungen angepasst.

Bei der Methode werden nicht Sprachmerkmale miteinander verglichen, da die Sprache nur eine Umsetzung eines zugrunde liegenden Modells ist. Vielmehr werden die zugrunde liegenden Modelle abgeleitet. Diese Modelle werden dann verglichen.

Die Methode selbst ist nicht auf die Beschreibungssprachen in der Automatisierungstechnik eingeschränkt, sondern funktioniert für jede Beschreibungssprache, die bei der Erstellung den beschriebenen Weg durchläuft.

Die Methode wurde dann auf marktgängige Gerätebeschreibungssprachen aus der Automatisierungstechnik angewendet. Hierbei zeigte sich, dass keine der betrachteten Sprachen eine umfassende funktionale Sicht auf die Geräte, die sie beschreiben soll, bietet.

Alle betrachteten Beschreibungssprachen bieten die Möglichkeit, Geräte für den Engineeringprozess bereitzustellen. Sie beruhen auf unterschiedlichen Sichten auf die Geräte. Während GDS und GSDML der Gedanke an modulare E/A-Geräte zugrunde liegt, bietet FDCML eine sehr viel generischere Sicht auf die Geräte. Tabelle 5.1 fasst die Bewertung zusammen.

Tabelle 5.1: Bewertung der betrachteten Gerätebeschreibungssprachen

	GSD	GSDML	FDCML	EDS
Darstellungsmöglichkeit von Antriebsgeräten	◐	◐	●	◐
Darstellungsmöglichkeit von E/A-Geräten	●	●	●	●
Darstellungsmöglichkeit von Hybridgeräten	◐	◐	●	◐
Darstellungsmöglichkeit von Funktionen	○	○	○	○
Darstellungsmöglichkeiten von Profilen	○	○	◐	○

● Erfüllt ◐ Teilweise erfüllt ○ Nicht erfüllt

Allen heutigen Lösungen ist aber gemein, dass sie keine Beschreibung von Profilen oder funktionalen Zusammenhängen erlauben. Somit ist ein funktionales Engineering nicht möglich.

Dieses Defizit gilt es nun in den folgenden Kapiteln zu beheben, indem ein Informationsmodell für die Automatisierungstechnik erarbeitet wird, welches eine funktionale Sicht auf Geräte erlaubt.

6 Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten

In diesem Kapitel wird ein Informationsmodell erarbeitet, das eine funktionale Typisierung von Automatisierungsgeräten ermöglicht. Hierzu werden zunächst mittels einer Analyse die Anforderungen an das Modell hergeleitet. In einem nächsten Schritt wird aufgrund der Ergebnisse der Anforderungsanalyse ein Konzept erstellt. Nach Erstellung des Konzeptes wird durch Informationsmodellierung das Modell abgeleitet.

6.1 Anforderungsanalyse

Ziel des Arbeitspunktes ist es, ein Informationsmodell zu schaffen, das bei der Entscheidung helfen kann, welche Geräte zum Ausführen einer bestimmten Applikation notwendig sind.

Um dieses Ziel zu erreichen, reichen heutige Beschreibungsformen für Geräte nicht aus, da sie es nicht erlauben, die durch ein Gerät bereitgestellten Funktionen abzubilden.

Anforderung an ein Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten ist, dass es ermöglicht, darzustellen, welche Funktionen durch ein Gerät erbracht werden können.

6.2 Konzeption

Bild 6.1 zeigt das grundsätzliche Konzept einer formal typisierten Gerätebeschreibung. Das Konzept beruht auf dem in 5.2 gezeigten Entstehungsprozess für eine Gerätebeschreibungssprache. Es soll die in 6.1 abgeleitete Anforderung erfüllen, einem Gerät zuordnen zu können, welche Funktionen es bereitstellt.

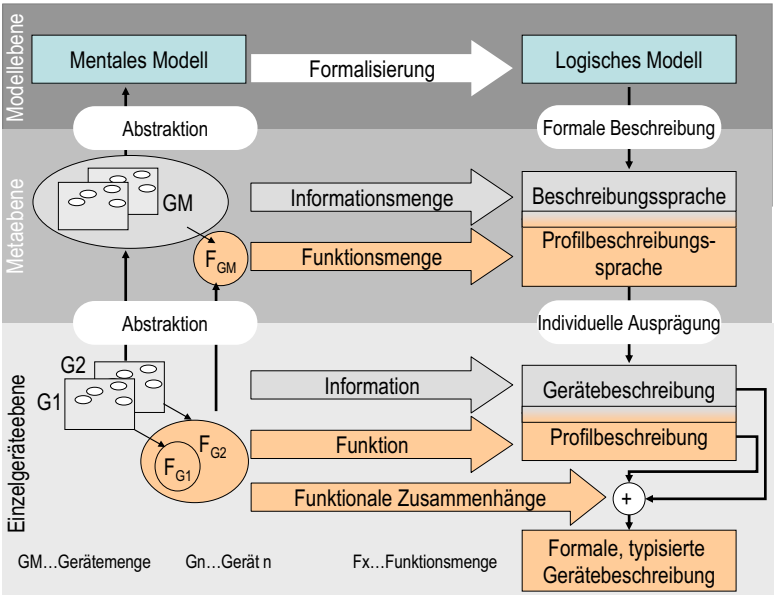


Bild 6.1: Konzept für eine formale, typisierte Gerätebeschreibung

Das Konzept wird dahingehend erweitert, dass nicht nur die Geräte, sondern auch die durch sie bereitgestellten Funktionen betrachtet werden. So stellt jedes Gerät Gn eine Funktionalität F_{Gn} bereit.

Die Funktionen, die durch die einzelnen Geräte bereitgestellt werden, können als Mengen dargestellt werden, aus denen sich Mengenrelationen zwischen den einzelnen Funktionen der Geräte ableiten lassen.

Auf der Metaebene wird aus den Funktionsmengen der Einzelgeräte die durch eine Gerätemenge GM bereitgestellte Funktionsmenge F_{GM}.

Das auf diese Weise angereicherte Mentale Modell kann wiederum durch Formalisierung in ein Logisches Modell überführt werden, welches damit auch eine Erweiterung gegenüber dem in 5.2 gezeigten besitzt. Der Unterschied besteht darin, dass das Logische Modell nun um funktionale Eigenschaften der Geräte angereichert ist.

Durch formale Beschreibung dieses Logischen Modells entsteht eine Beschreibungsform, die neben der Beschreibung der Geräte auch eine Beschreibung von Funktionen und ihrer Zusammenhänge erlaubt.

Durch individuelle Ausprägung ergibt sich neben der Gerätebeschreibung auch die Möglichkeit einer Funktionsbeschreibung und einer Klassifizierung von Funktionen. Werden diese beiden verknüpft, so ist eine formale, typisierte Gerätebeschreibung entstanden, die es erlaubt, funktionale Zusammenhänge von Geräten zu beschreiben.

6.2.1 Definition eines allgemeingültigen Gerätemodells

Ein Gerät, wie es in 2.9 eingeführt wurde, ist eine abgeschlossene Komponente, die unterschiedliche Funktionen anbietet.

Der abstrakte Aufbau eines Gerätes ist in Bild 6.2 gezeigt. Hierbei wird ein modularer Aufbau gewählt. Ein Gerät kann aus mehren Teilen bestehen. Diese Teile werden Module genannt. Jedes Modul kann wiederum aus einem Hardwareteil und einem Teil, der in Software realisiert ist, bestehen. Diese Module können physikalisch im Gerät vorhanden sein oder auch nur der logischen Strukturierung des Gerätes dienen. Lassen sich Hard- oder Softwareteile nicht konkreten Modulen zuordnen, so werden sie dem Gerät selber zugeordnet. Beispiele hierfür sind die Kommunikationsschnittstelle des Gerätes auf Hardwareebene oder die Kommunikationssoftware des Gerätes, die beide durch das gesamte Gerät genutzt werden.

Sinnvollerweise sind diese Module wiederum Komponenten, das heißt sie sind Einheiten, die beim Hersteller des Gerätes gesondert bestellbar sind. Beispiele hierfür sind gerätetechnische Module eines modularen Eingangs-/Ausgangssystems, ein Motor eines Antriebssystems oder ein Optionsmodul einer Firmware, die eine bestimmte Funktionalität zusätzlich frei schaltet. Anhand der Beispiele wird klar, dass ein Modul nicht immer Hard- und Software enthalten muss.

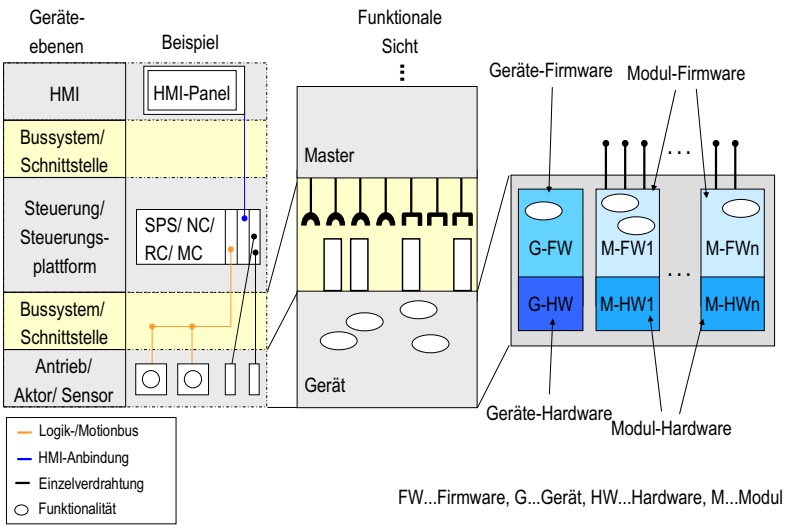


Bild 6.2: Aufbau eines Gerätes

6.2.2 Einordnen von Metaprofilen in das logische Modell

Um die Aufteilung der Geräte in die in 6.2.1 beschriebenen Teile „Modul“ und „Gerät“ zu vereinfachen und somit auch eine Klassifikation der Geräte nach funktionalen Gesichtspunkten zu erreichen, werden den einzelnen Teilen innerhalb des abstrakten Gerätes unterschiedliche Metaprofile zugeordnet.

Ein Metaprofil stellt in diesem Zusammenhang eine Gruppe von Profilen da, die funktional auf derselben Ebene des OSI-Referenzmodells zu finden sind. Im Rahmen der Automatisierungstechnik sind typischerweise nur Anwendungsschicht, Sicherungsschicht und Bitübertragungsschicht relevant, wie schon in 2.2 beschrieben wurde. Darüber hinaus wird über der Anwendungsschicht den Kommunikationsobjekten in einer weiteren Ebene eine Semantik gegeben, wie es schon in Bild 2.3 beispielhaft dargestellt worden ist.

Die Anwendungsschicht, die Sicherungsschicht und die Bitübertragungsschicht werden in einem Metaprofil zusammengefasst, da sie zusammen die Kommunikationsprotokolle

beschreiben, mit denen ein Gerät angesprochen werden kann. Diese Eigenschaften sind dem gesamten Gerät zugehörig und sind typischerweise in der Firmware des Gerätes abgebildet, wie auch in Bild 6.3 zu sehen ist.

Die Ebene über der Anwendungsschicht, die den Kommunikationsobjekten eine Semantik gibt, wird ein weiteres Metaprofil. In diesem werden die Applikationsprofile zusammengefasst. Die Funktionen, die von bestimmten Applikationen benötigt werden, finden sich in den einzelnen Modulen innerhalb des Gerätes. Somit implementieren diese Teile jeweils Applikationsprofile.

Das Gerät besitzt bestimmte physikalische Eigenschaften, wie beispielsweise Anschlüsse für Kommunikation oder Stromversorgung. Diese Eigenschaften werden in Gerätehardwareprofilen zusammengefasst.

Auch einzelne Module können dedizierte Hardware beinhalten. Diese wird dann den Modulhardwareprofilen zugeordnet.

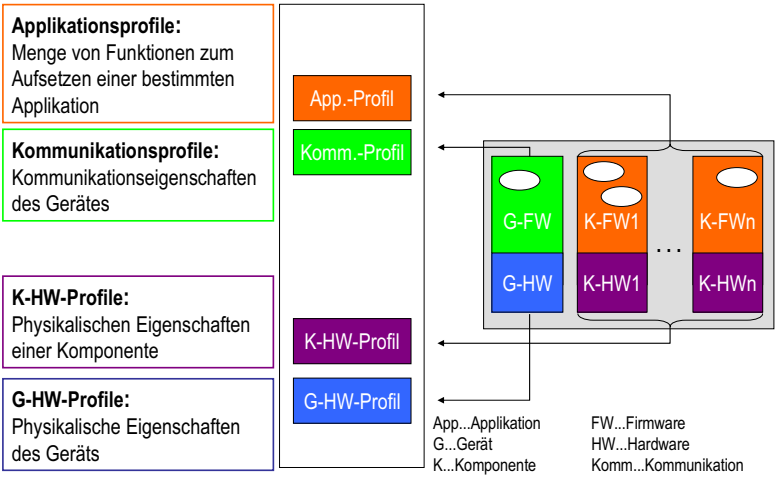


Bild 6.3: Profile im logischen Modell

6.2.3 Aufbau des Gerätemodells

Aus den Festlegungen in 6.2.2 folgt das Gerätemodell wie es in Bild 6.4 dargestellt ist. Ein Gerät hat jeweils gerätespezifische Hardware und gerätespezifische Firmware.

Ein Gerät besteht zusätzlich aus mindestens einem Modul. Ein Modul besitzt eine funktionale Repräsentanz in der Firmware und kann zusätzlich auch noch eine modulspezifische Hardware haben.

Die Modulfirmware implementiert dabei die Funktionen, die für das Aufsetzen einer bestimmten Applikation in dem Gerät notwendig ist. Damit wird kann die Modulfirmware durch ein oder mehrere Applikationsprofile beschrieben werden. Die Modulhardware lässt sich durch ein Modulhardwareprofil beschreiben.

Die Gerätefirmware implementiert die kommunikativen Funktionen des Gerätes und legt somit die für die Applikationsprofile aller Module möglichen Kommunikations-mechanismen fest.

Auf diese Art ist eine Klassifikation der Geräte auf mehreren Ebenen möglich. Aus den einzelnen Beziehungen untereinander lässt sich ein Gerät folgendermaßen beschreiben:

- Ein Gerät wird durch genau ein Kommunikationsprofil beschrieben.
- Ein Gerät implementiert ein oder mehrere Applikationsprofile.
- Ein Gerät kann mehrere Modulhardwareprofile haben, muss es aber nicht. Die Anzahl der in einem Gerät vorhandenen Modulhardwareprofile ist kleiner oder gleich der Anzahl der in dem Gerät vorhandenen Module.
- Ein Gerät hat genau ein Gerätehardwareprofil.

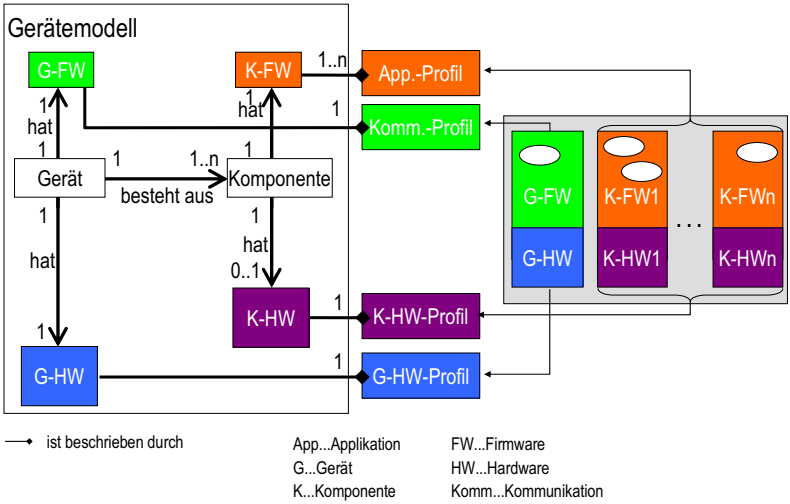


Bild 6.4: Gerätemodell

6.3 Zusammenfassung

In diesem Kapitel wurde ein Gerätemodell abgeleitet, welches eine funktionale Sicht auf Automatisierungsgeräte erlaubt. Das Modell benutzt hierzu Profile auf mehreren Ebenen, die sich am OSI-Referenzmodell orientieren.

Ein Gerät besitzt demnach aus funktioneller Sicht ein Kommunikationsprofil, welches die kommunikativen Eigenschaften des Gerätes abbildet und ein oder mehrere Applikationsprofile, welche die eigentliche Funktion des Gerätes bereitstellen. Die Hardware des Gerätes und der Module wird durch Hardwareprofile beschrieben.

7 Funktionale Klassifikation von Automatisierungsgeräten

Um eine funktionale Klassifikation von Automatisierungsgeräten zu erreichen, wird in diesem Kapitel eine Methode erarbeitet, die eine funktionale Gruppierung von Kommunikationsobjekten erlaubt. Um diese Methode abzuleiten, werden Aspekte des objektorientieren Entwurfs und der Graphentheorie verwendet.

7.1 Funktionale Profile für Automatisierungsgeräte

Automatisierungsgeräte lassen sich nach ihren funktionalen Schnittstellen klassifizieren. Die implementierten Schnittstellen eines Gerätes bilden somit die Grundlage für die Klassifizierung.

7.1.1 Notwendigkeit für funktionale Profile

Um eine Klassifizierung nach den funktionalen Eigenschaften zu ermöglichen, werden die funktionalen Schnittstellen der zu betrachtenden Geräte in Gruppen eingeteilt. Diese Gruppen werden Profile genannt.

7.1.2 Strukturierung von Profilen

Innerhalb der Profile existieren wiederum Gruppierungsmöglichkeiten. Es wird eine Strukturierung nach dem OSI-Referenzmodell gewählt. Dies ermöglicht eine durchgängige funktionale Klassifizierung von Automatisierungsgeräten. Dieser Ansatz erlaubt es, auf Applikationsebene Geräte in Klassen zusammenzufassen, obwohl sie unterschiedliche Kommunikationsklassen implementieren.

Deswegen werden zunächst zwei Metaprofile eingeführt. Eines fasst die Kommunikationseigenschaften des Gerätes zusammen. Das andere gruppiert die Applikationseigenschaften des Gerätes. Das Vorgehen ist in Bild 7.1 dargestellt.

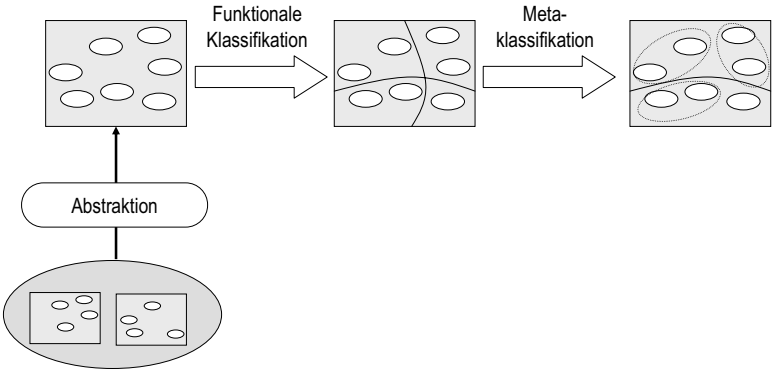


Bild 7.1: Funktionale Klassifikation von Geräten

Bei der Klassifikation ergibt sich, dass auf Applikationsebene eine weitere Klassifizierung notwendig ist, um Mehrfachvererbung auf dieser Ebene zu vermeiden. So besitzen Geräte auf Applikationsebene Funktionen wie beispielsweise Diagnose, die nicht direkt mit der eigentlichen Funktion wie zum Beispiel Antriebsfunktion oder E/A-Funktion gekoppelt sind. Deswegen wird das Applikationsmetaprofil nochmals in zwei Metaprofile unterteilt. Eines dient zur Gruppierung der Verwaltungsfunktionen der Geräte und das andere beinhaltet alle Profile, die mit der applikativen Funktion des Gerätes betraut sind. Das erste gruppiert die generischen Applikationsfunktionen, das andere die funktionsspezifischen Applikationsfunktionen.

7.2 Vorgehen zur funktionalen Klassifikation von Automatisierungsgeräten

Bild 7.2 zeigt den grundsätzlichen Ablauf zum Aufbau einer funktionalen Klassifikation von Automatisierungsgeräten in Geräteklassen. Dieser Ablauf wird in einer ersten Iteration für eine grobe Klassifizierung der Geräte und zur Ableitung bestehender Abhängigkeiten von Kommunikationsobjekten untereinander verwendet.

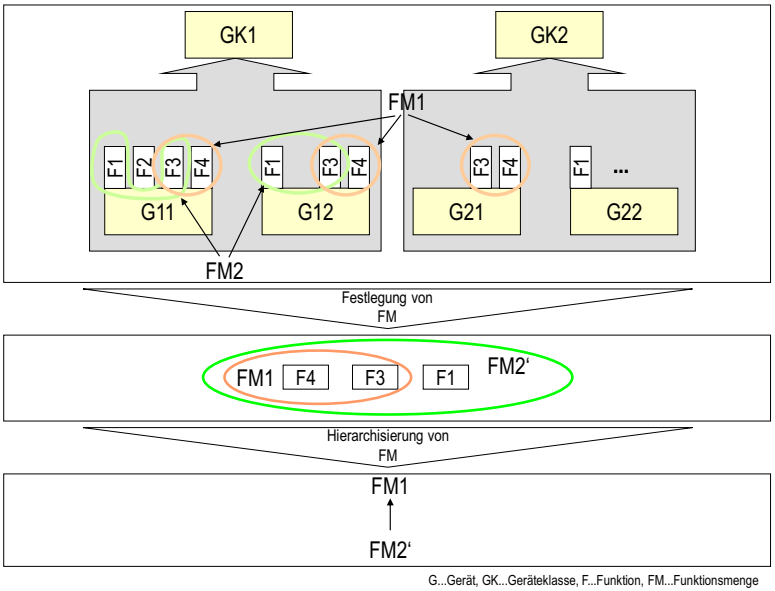


Bild 7.2: Ablauf der funktionalen Klassifikation

In einem ersten Schritt werden einzelne Geräteklassen und deren Repräsentanten betrachtet. Diese erste Geräteklassifikation kann beispielsweise aus dem Sprachgebrauch oder einem Produktkatalog abgeleitet werden. So ist hier eine Klassifikation beispielsweise in Antriebsgeräte, Eingangs-Ausgangsgeräte und Hybridgeräte denkbar.

Die Funktionen der betrachteten Geräte werden dann zu Funktionsmengen zusammengefasst. Da die Funktionen in Form von Kommunikationsobjekten bereitgestellt werden, entspricht dieses Vorgehen auch einer Gruppierung von Kommunikationsobjekten.

Besteht zum Beispiel im Rahmen einer Norm oder sonstigen Richtlinie bereits eine Gruppierungsmöglichkeit von Kommunikationsobjekten, die durch die Geräte bereitgestellt werden, so kann die erste Gruppierung nach dieser geschehen.

Ziel ist, dass die gefundenen Mengen entweder jeweils keine gleichen Elemente besitzen, also disjunkt sind, oder echte Teilmengen voneinander sind. Auf diese Weise wird eine Mehrfachvererbung innerhalb der durch die Funktionen festgelegten

Schnittstellen in den Geräten vermieden. Eine zweite Randbedingung ist, dass Schnittstellen aller Geräte direkt durch die Kombination von Implementierungen einzelner Funktionsmengen darstellbar sind.

Es ist möglich, dass die erste Gruppierungsmöglichkeit der Funktionen den Zielen und Randbedingungen nicht entspricht. Dann müssen die Funktionen neu strukturiert werden. Eine Methode zum Erreichen dieses Ziels wird im folgenden Kapitel hergeleitet und beschrieben.

Sind die Gruppen von Kommunikationsobjekten zur funktionalen Typisierung gefunden, so können diese hierarchisiert werden. Aufgrund dieser Hierarchisierung können dann auch die Geräteklasse funktional neu geordnet werden, indem sich die neuen Geräteklassen an den Abhängigkeiten der funktionalen Gruppen ausrichten.

7.3 Methode zum Festlegen vom Mengen von Kommunikationsobjekten

Im Folgenden wird eine Methode hergeleitet, die es erlaubt, aus den Beziehungen von Kommunikationsobjekten untereinander Funktionsmengen abzuleiten. Diese Methode kann auch in Form eines Programms umgesetzt werden.

7.3.1 Fähigkeiten von Kommunikationsobjekten

Einzelne Kommunikationsobjekte, wie sie in Kapitel 2.11 eingeführt wurden, haben unterschiedliche Fähigkeiten und statische Eigenschaften. Diese beziehen sich auf das reale Objekt selbst und nicht auf das abstrakte Objekt, von dem das reale Objekt abgeleitet ist. Das abstrakte Objekt wird in Form einer Spezifikation oder einer sonstigen Richtlinie beschrieben. Hierbei kann es sich um die Beschreibung eines Einzelobjektes oder einer Menge von Einzelobjekten handeln.

Ein durch eine Spezifikation beschriebenes abstraktes Einzelobjekt kann beispielsweise einen bestimmten Wertebereich für einen zugreifbaren Datensatz vorgeben, die Umsetzung in einem Gerät kann diesen Wertebereich aber weiter einschränken.

Als mögliche Fähigkeits- bzw. Eigenschaftsdimensionen eines Kommunikationsobjektes können beispielsweise gelten:

- **Zugriffsmöglichkeiten**
  Innerhalb der Dimension **Zugriffsmöglichkeiten** sind folgende Ebenen möglich:
  - Einzelne Teile des Kommunikationsobjektes sind von außen über azyklische Kommunikation lesbar.
  - Einzelne Teile des Kommunikationsobjektes sind von außen über zyklische Kommunikation lesbar.
- **Änderungsmöglichkeiten**
  Innerhalb der Dimension **Änderungsmöglichkeiten** sind folgende Ebenen möglich:
  - Es besteht keine Änderbarkeit von Teilen des Kommunikationsobjektes.
  - Einzelne Teile des Kommunikationsobjektes sind durch das Gerät selbst änderbar.
  - Einzelne Teile des Kommunikationsobjektes sind von außen über azyklische Kommunikation änderbar.
  - Einzelne Teile des Kommunikationsobjektes sind von außen über zyklische Kommunikation änderbar.
- **Wertebereich**
  Innerhalb der Dimension **Wertebereich** sind pro Attribut des Kommunikationsobjektes folgende Ebenen möglich:
  - Es existiert ein linearer möglicher Wertebereich, das heißt die Werte liegen zwischen einem definierten Minimum und einem definierten Maximum.
  - Es existiert ein mehrdimensionaler Wertebereich. Dies ist für sogenannte Bitlisten relevant. In diesen Bitlisten werden unterschiedliche Eigenschaften über einzelne Bits oder Bitkombinationen angezeigt. Die einzelnen Teile einer Bitliste haben nicht zwingend eine Abhängigkeit untereinander.
- **Instanziierbarkeit**
  Innerhalb der Dimension **Instanziierbarkeit** sind folgende Ebenen möglich:
  - Das Kommunikationsobjekt ist genau n-mal vorhanden.
  - Das Kommunikationsobjekt ist mindestens n-mal, höchstens aber m-mal instanziierbar.

7.3.2 Beziehungen zwischen Fähigkeiten von Kommunikationsobjekten

7.3.2.1 Einfache Beziehungen

Innerhalb einer Spezifikation können verschiedene Kommunikationsobjekte unterschiedliche Beziehungen zueinander haben, so können einzelne Kommunikationsobjekte unterschiedlich interagieren.

Für die Gruppierung von solchen Objekten reicht allerdings eine Art von Beziehung aus, nach der die Objekte zusammengefasst werden sollen.

In der hier beschriebenen Methode werden Kommunikationsobjekte in Verbindung zueinander gesetzt. Dadurch entsteht ein Graph, in dem die Kommunikationsobjekte als Knoten auftauchen. Die Kanten repräsentieren die gerichtete Abhängigkeit der beteiligten Knoten, Dieser Graph soll Aussagen über Gruppierbarkeit von Kommunikationsobjekten innerhalb des Graphen ermöglichen. Der entstehende Graph ist ein endlicher gerichteter Multigraph, wie er in 4.5 beschrieben ist, da die einzelnen Knoten durch mehrere gerichtete Kanten verbunden werden können.

Die einfachste Beziehung zweier Knoten innerhalb dieses Graphen ist in Bild 7.3 dargestellt. Die Abbildung zeigt eine einfache Abhängigkeit zweier Kommunikationsobjekte A und B. Der Pfeil, der die beiden Knoten verbindet, bedeutet in Prädikatenlogik: Das Vorhandensein von KO A impliziert das Vorhandensein von KO B. Somit ist KO B immer vorhanden, sobald KO A vorhanden ist. Das Vorhandensein von KO A ist also im mathematischen Sinne hinreichend für das Vorhandensein von KO B. Umgekehrt ist das Vorhandensein von KO B in mathematischen Sinn notwendig für das Vorhandensein von KO A. In mathematischer Notation wird dieser Zusammenhang folgendermaßen ausgedrückt: $A \Rightarrow B$.

Der Graph $G = (V, E)$ mit $V = \{A, B\}$ stellt sich in diesem Fall dar als

$$G = (\{A, B\}, \{\overrightarrow{\mathrm{AB}}\}) \tag{7.1}$$

oder in Matrixschreibweise mittels der Adjazenzmatrix

$$G(\{A,B\}) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{7.2}$$

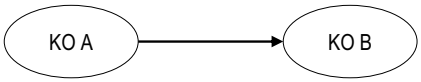


Bild 7.3: Einfache, gerichtete Abhängigkeit zweier Kommunikationsobjekte

In Bild 7.4 ist eine gegenseitige Abhängigkeit zwischen zwei Kommunikationsobjekten dargestellt. In Prädikatenlogik bedeutet diese Darstellung: Das Vorhandensein von KO A impliziert das Vorhandensein von KO B und das Vorhandensein von KO B impliziert das Vorhandensein von KO A. KO A ist genau dann vorhanden, wenn KO B auch vorhanden ist. Das Vorhandensein von KO A ist also notwendig und hinreichend für das Vorhandensein von KO B. In mathematischer Notation wird dieser Zusammenhang folgendermaßen ausgedrückt: $A \Leftrightarrow B$.

Der Graph $G = (V, E)$ mit $V = \{A, B\}$ stellt sich in diesem Fall dar als

$$G = (\{A,B\}, \{\overrightarrow{AB}, \overrightarrow{BA}\}) \tag{7.3}$$

oder in Matrixschreibweise mittels der Adjazenzmatrix

$$G(\{A,B\}) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{7.4}$$

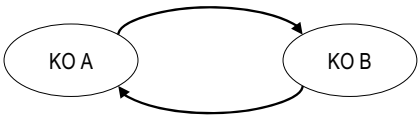


Bild 7.4: Gegenseitige Abhängigkeit zweier Kommunikationsobjekte

In Bild 7.5 ist eine zyklische Abhängigkeit von drei Kommunikationsobjekten dargestellt. Im Rahmen der Spezifikation wurden für diese drei Kommunikationsobjekte folgende Zusammenhänge aufgestellt:

- Das Vorhandensein von KO A impliziert das Vorhandensein von KO B. Mathematisch: $A \Rightarrow B$
- Das Vorhandensein von KO B impliziert das Vorhandensein von KO C. Mathematisch: $B \Rightarrow C$
- Das Vorhandensein von KO C impliziert das Vorhandensein von KO A. Mathematisch: $C \Rightarrow A$

Der Graph $G = (V, E)$ mit $V = \{A, B, C\}$ stellt sich in diesem Fall dar als

$$G = (\{A, B, C\}, \{\overrightarrow{AB}, \overrightarrow{BC}, \overrightarrow{CA}\}) \tag{7.5}$$

oder in Matrixschreibweise mittels der Adjazenzmatrix

$$G(\{A, B, C\}) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{7.6}$$

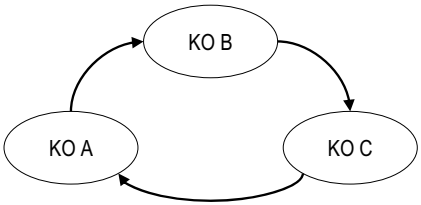


Bild 7.5: Zyklische Abhängigkeit bei drei Kommunikationsobjekten

Da $A \Rightarrow B$ und $B \Rightarrow C$ folgt $A \Rightarrow C$. Mittels dieser Logik lassen sich folgende Aussagen ableiten:

- Das Vorhandensein von KO A impliziert das Vorhandensein von KO C. Mathematisch wird diese Aussage wie folgt beschrieben: $A \Rightarrow C$. Da aber gilt: $C \Rightarrow A$, folgt: $A \Leftrightarrow C$.
- Das Vorhandensein von KO B impliziert das Vorhandensein von KO A. Mathematisch wird diese Aussage wie folgt beschrieben: $B \Rightarrow A$. Da aber gilt: $A \Rightarrow B$, folgt: $A \Leftrightarrow B$.
- Das Vorhandensein von KO C impliziert das Vorhandensein von KO B. Mathematisch wird diese Aussage wie folgt beschrieben: $C \Rightarrow B$. Da aber gilt: $B \Rightarrow C$, folgt: $B \Leftrightarrow C$.

Der Graph $G' = (V, E)$ mit $V = \{A, B, C\}$ stellt sich in diesem Fall dar als

$$G' = (\{A, B, C\}, \{\overrightarrow{AB}, \overrightarrow{AC}, \overrightarrow{BA}, \overrightarrow{BC}, \overrightarrow{CA}, \overrightarrow{CB}\}) \tag{7.7}$$

oder in Matrixschreibweise mittels der Adjazenzmatrix

$$G'(\{A, B, C\}) = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \tag{7.8}$$

$$G'(\{A, B, C\}) = G \vee G^2 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \vee \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{7.9}$$

$G'(\{A, B, C\})$ verbindet also die Möglichkeiten, die sich durch zweimaliges Anwenden von G ergeben.

Somit lässt sich Bild 7.5 direkt in Bild 7.6 umschreiben und die beiden Abbildungen sind im Bezug auf die logische Aussage gleichbedeutend.

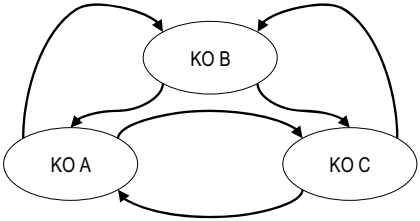


Bild 7.6: Zyklische Abhängigkeit bei drei Kommunikationsobjekten

Es ergibt sich:

$$G^3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.10}$$

Somit ist nach dreimaligem Ausführen von G jeweils wieder der Ausgangspunkt erreicht. Der Graph bildet also ein wie in Bild 7.5 zu sehen ist eine zyklische Abhängigkeit von drei Objekten in der dritten Iteration der Transitivitäten.

Durch die Summe der drei ersten Potenzen von $G(\{A,B,C\})$ ergibt sich die sogenannte transitive Hülle von $G(\{A,B,C\})$. Diese wird mathematisch mit $G^+(\{A,B,C\})$ bezeichnet.

Im vorliegenden Fall ist

$$G^{+}(\{A,B,C\}) = \bigcup_{i \in N} G^{i}(\{A,B,C\}) = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{7.11}$$

Es folgt also innerhalb des geschlossenen Zyklus, dass alle Elemente sich gegenseitig und auch selbst implizieren, sobald eines vorhanden ist.

Was hier für die zyklische Abhängigkeit von drei Kommunikationsobjekten gezeigt wurde, lässt sich leicht durch vollständige Induktion für jede Anzahl von Kommunikationsobjekten innerhalb einer zyklischen Abhängigkeit ableiten.

Somit folgt, dass sobald sich in einem gerichteten Graphen eine Verbindung zwischen zwei Kommunikationsobjekten K_0 und K_m aufbauen lässt, die Abhängigkeitspfeilen folgt, für alle Knoten entlang dieser Verbindung gilt:

$$K_a \Rightarrow K_b \forall a < b \tag{7.12}$$

In der Matrixdarstellung der transitiven Hülle G^{+} des Graphen G sind also Einträge an jeder Stelle $\{a,b\}$ vorhanden.

Es folgt außerdem, dass sobald sich in einem gerichteten Graphen eine Masche aufbauen lässt, die Abhängigkeitspfeilen folgt, für alle Knoten entlang dieser Masche wechselseitig gilt:

$$K_a \Leftrightarrow K_b \forall a,b \tag{7.13}$$

In der Matrixdarstellung der transitiven Hülle G^{+} des Graphen G sind also Einträge an jeder Stelle $\{a,b\}$ und $\{b,a\}$ vorhanden. Dies gilt auch für den Fall $a = b$. Ein Element, das in der transitiven Hülle G^{+} des Graphen G einen Eintrag in der Hauptdiagonalen hat, ist also Teil einer Masche.

7.3.2.2 Optionale und verbindliche Kommunikationsobjekte

Innerhalb von Spezifikationen gibt es typischerweise zwei unterschiedliche Ebenen von Kommunikationsobjekten, optional und verbindlich vorhandene. Dies soll auch im Abhängigkeitsgraphen dargestellt werden.

Hierzu wird folgende Notation eingeführt. Es sei K ein verbindlich vorhandenes Kommunikationsobjekt:

$$\overline{\mathrm{K}} :\Leftrightarrow \emptyset \Leftrightarrow K \tag{7.14}$$

Es wird hierfür in der Matrix eine weitere Spalte und Zeile für die leere Menge eingeführt. In Matrixschreibweise wird dies durch den Wert eins an der Stelle $\{0, K\}$ des Graphen $G(V, E)$ ausgedrückt.

Innerhalb eines Graphen soll diese Information wie in Bild 7.7 gezeigt dargestellt werden.

Bild 7.7: Verbindlich vorhandenes Kommunikationsobjekt

Wenn nun das Vorhandensein eines verbindlichen Kommunikationsobjektes K_m das Vorhandensein eines anderen Kommunikationsobjektes impliziert, so lässt sich schreiben:

$$\overline{\mathrm{K}_{\mathrm{m}}} \Rightarrow K_l \tag{7.15}$$

Aus der Definition der Verbindlichkeit und unter Zuhilfenahme der Abhängigkeitsprogression aus 7.3.2.1 lässt sich schreiben:

$$\emptyset \Rightarrow K_l \tag{7.16}$$

Für jede Menge K_l gilt:

$$K_l \Rightarrow \emptyset \tag{7.17}$$

Somit kann folgendes geschrieben werden:

$$\emptyset \Leftrightarrow K_l \tag{7.18}$$

Für verbindlich vorhandene Kommunikationsobjekte gilt also, dass sie diese Eigenschaft entlang einer gerichteten Verbindung weitergeben. Es gilt als:

$$(\overline{\mathrm{K_m}} \Rightarrow K_l) \Rightarrow \overline{\mathrm{K_l}} \tag{7.19}$$

Für jedes verbindlich vorhandene Kommunikationsobjekt K_i gilt:

$$\emptyset \Leftrightarrow K_i \tag{7.20}$$

Es lässt sich also über die leere Menge eine Masche zwischen allen verbindlich vorhandenen Kommunikationsobjekten schließen. Mittels der in 7.3.2.1 abgeleiteten Folgerung bezüglich der Abhängigkeiten innerhalb einer Masche lässt sich somit für alle verbindlich vorhandenen Kommunikationsobjekte schreiben:

$$\overline{K_a} \Leftrightarrow \overline{K_b} \forall a,b \tag{7.21}$$

7.3.2.3 Komplexere Beziehungen

Über die einfache Beziehung, die in 7.3.2.1 beschrieben und verwendet wurde, gibt es in Spezifikationen auch noch eine komplexere Beziehung von Kommunikationsobjekten bzw. deren Fähigkeiten und Eigenschaften, nämlich den gegenseitigen Ausschluss des Vorhandenseins.

In Bild 7.8 ist ein solcher Zusammenhang dargestellt. In Prädikatenlogik bedeutet dieses Bild: Das Vorhandensein von KO A impliziert das Nichtvorhandensein von KO B und umgekehrt. Das Vorhandensein von KO A und KO B schließen sich gegenseitig aus. Mathematisch ist das Nichtvorhandensein eines KOs notwendig, aber keinesfalls hinreichend für das Vorhandensein des anderen, da auch beide nicht vorhanden sein können. Es gilt somit:

$$A \Rightarrow \neg B \text{ und } B \Rightarrow \neg A \tag{7.22}$$

Der Graph $G = (V, E)$ mit $V = \{A, B\}$ stellt sich in diesem Fall dar als

$$G = \{(A, B), (B, A)\} \tag{7.23}$$

oder in Matrixschreibweise als

$$G(\{A, B\}) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{7.24}$$

Hierbei ist zu beachten, dass für den Graphen $G = (V, E)$, der den gegenseitigen Ausschluss von Kommunikationsobjekten beschreibt, gilt:

$$G = G^T \tag{7.25}$$

Die Matrixdarstellung von G und ihrer Transponierten G^T sind identisch. Sie ist also symmetrisch bezüglich der Hauptdiagonalen. Um keinen Widerspruch in sich zu haben, dürfen keine Einträge auf der Hauptdiagonalen existieren, da sich sonst ein Kommunikationsobjekt selbst ausschließt.

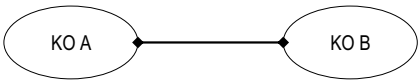


Bild 7.8: Gegenseitiger Ausschluss von Kommunikationsobjekten

Mit den in 7.3.2.1 abgeleiteten Sätzen bezüglich der Abhängigkeiten von Kommunikationsobjekten entlang einer Verbindung bzw. innerhalb einer Masche eines Graphen lassen sich mit der eingeführten Notation des gegenseitigen Ausschließens Unstimmigkeiten in dem Graphen erkennen.

Bild 7.9 zeigt einen möglichen Ausschnitt eines Graphen mit dem gegenseitigen Ausschluss zweier Kommunikationsobjekte. Es besteht kein logischer Widerspruch in dem Graphen:

1. Das Vorhandensein von KO A impliziert das Vorhandensein von KO C.

2. Das Vorhandensein von KO C impliziert das Vorhandensein von KO D.
3. Das Vorhandensein von KO B impliziert das Vorhandensein von KO D.
4. Das Vorhandensein von KO A und das Vorhandensein von KO B schließen sich gegenseitig aus.
5. Aus 1 und 2 folgt: Das Vorhandensein von KO A impliziert das Vorhandensein von KO D.

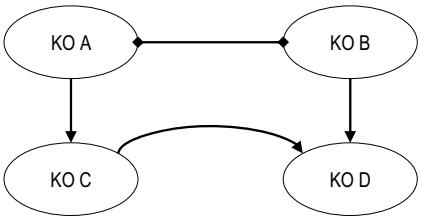


Bild 7.9: Möglicher Abhängigkeitsgraph mit Ausschluss zweier Objekte

Es ergeben sich zwei überlagerte Graphen $G_1 = (V, E)$ und $G_2 = (V, E)$. Der Graph $G_1 = (V, E)$ mit $V = \{A, B, C, D\}$, der den gegenseitigen Ausschluss von A und B darstellt, stellt sich in diesem Fall dar als

$$G_1 = \{(A, B), (B, A)\} \tag{7.26}$$

oder in Matrixschreibweise als

$$G_1(\{A, B, C, D\}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.27}$$

und der $G_2 = (V, E)$ mit $V = \{A, B, C, D\}$, der die Abhängigkeiten des Vorhandenseins der Kommunikationsobjekte darstellt, stellt sich in diesem Fall dar als

$$G_2 = \{(A, C), (C, D), (B, C)\} \tag{7.28}$$

oder in Matrixschreibweise als

$$G_2(\{A,B,C,D\}) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.29}$$

Die transitive Hülle von $G_2 = (V,E)$ ergibt sich zu

$$G_2^+(\{A,B,C,D\}) = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.30}$$

Werden nun $G_2^+(\{A,B,C,D\})$ und $G_1(\{A,B,C,D\})$ verglichen, ergibt sich kein Widerspruch, da es keine Stellen in den Matrizen gibt, die in beiden Matrizen besetzt sind.

Anders verhält es sich in Bild 7.10. Die Abbildung zeigt ein Beispiel, bei dem es zu einem logischen Widerspruch kommt:

1. Das Vorhandensein von KO A impliziert das Vorhandensein von KO C.
2. Das Vorhandensein von KO C impliziert das Vorhandensein von KO D.
3. Das Vorhandensein von KO D impliziert das Vorhandensein von KO B.
4. Das Vorhandensein von KO A und das Vorhandensein von KO B schließen sich gegenseitig aus.
5. Aus 1, 2 und 3 folgt: Das Vorhandensein von KO A impliziert das Vorhandensein von KO B.

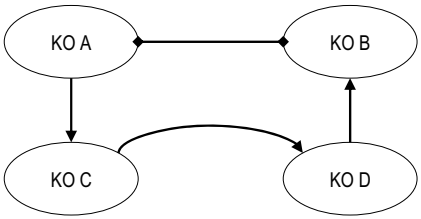


Bild 7.10: Unmöglicher Abhängigkeitsgraph mit Ausschluss zweier Objekte

Die Aussagen 4 und 5 widersprechen sich, es liegt also ein Widerspruch in dem Graph oder sogar in der zugrunde liegenden Spezifikation vor.

Es ergeben sich zwei überlagerte Graphen $G_1 = (V, E)$ und $G_2 = (V, E)$. Der Graph $G_1 = (V, E)$ mit $V = \{A, B, C, D\}$, der den gegenseitigen Ausschluss von A und B darstellt, stellt sich in diesem Fall dar als

$$G_1 = \{(A, B), (B, A)\} \tag{7.31}$$

oder in Matrixschreibweise als

$$G_1(\{A, B, C, D\}) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.32}$$

und der $G_2 = (V, E)$ mit $V = \{A, B, C, D\}$, der die Abhängigkeiten des Vorhandenseins der Kommunikationsobjekte darstellt, stellt sich in diesem Fall dar als

$$G_2 = \{(A, C), (C, D), (B, C)\} \tag{7.33}$$

oder in Matrixschreibweise als

$$G_2(\{A,B,C,D\}) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{7.34}$$

Die transitive Hülle von $G_2 = (V,E)$ ergibt sich zu

$$G_2^+(\{A,B,C,D\}) = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{7.35}$$

Werden nun $G_2^+(\{A,B,C,D\})$ und $G_1(\{A,B,C,D\})$ verglichen, ergibt sich ein Widerspruch, da in beiden Matrizen die Stelle (A,B) besetzt ist.

Somit folgt, dass sobald sich in einem gerichteten Graphen eine Verbindung zwischen zwei Kommunikationsobjekten K_0 und K_m aufbauen lässt, die Abhängigkeitspfeilen folgt, und K_0 und K_m sich gegenseitig ausschließen, ein Widerspruch in der Logik der Abhängigkeiten vorhanden ist. Dies lässt darauf schließen, dass entweder der Graph falsch erstellt wurde oder in der zugrunde liegenden Spezifikation ein Fehler besteht.

Ein möglicher Fehler in der Spezifikation oder bei der Erstellung des Graphen lässt sich nachprüfen, indem zwei Matrizen G_1 und G_2 aufgestellt werden. Die Matrix G_1 stellt den gegenseitigen Ausschluss der Kommunikationsobjekte dar, G_2 die Abhängigkeiten des Vorhandenseins. Nun wird die transitive Hülle von G_2 G_2^+ gebildet und mit G_1 verglichen. Gibt es Stellen in diesen beiden Matrizen, die in beiden besetzt sind, so existieren Widersprüche.

7.3.3 Anwendungsfälle für einen Abhängigkeitsgraphen von Kommunikationsobjekten

Über den Anwendungsfall des Testes der Spezifikation wie er in 7.3.2.2 beschrieben wird, gibt es weiterhin mehrere Punkte im Lebenszyklus einer Automatisierungsanlage und der dazugehörigen Komponenten, bei der eine Gruppierung von

Kommunikationsobjekten über einen Graphen, der die Abhängigkeiten der Objekte untereinander darstellt, hilfreich ist.

Bei der Erstellung neuer Geräte kann dies beispielsweise hilfreich sein, da der Entwickler der Firmware des Gerätes aus diesem Graphen ableiten kann, welche Kommunikationsobjekte unter welchen Randbedingungen zu implementieren sind.

Wird beispielsweise eine bestimmte Funktionalität im Gerät benötigt und soll diese als Funktion über die Kommunikationsschnittstelle nach außen gegeben werden, so kann dies zu Folge haben, dass auch andere Funktionalitäten als Funktionen nach außen gegeben werden müssen, um eine Konformität mit der entsprechenden Norm zu gewährleisten.

Ein anderer Anwendungsfall ist die Planung einer Anlage. Soll durch die Anlage eine Funktion eines Gerätes genutzt werden, so kann sobald diese Funktion im Gerät vorhanden ist, auch auf das Vorhandensein anderer Funktionen geschlossen werden.

Ein dritter Fall ist ein automatischer Test im Rahmen einer Zertifizierung eines Gerätes. Das Testwerkzeug kann mittels des Abhängigkeitsgraphen automatisiert Testfälle erstellen, indem der Graph entlang der beschriebenen Abhängigkeiten das Vorhandensein der entsprechenden Kommunikationsobjekte und deren Funktionsebenen testet.

7.3.4 Regeln für die Gruppierbarkeit von Kommunikationsobjekten

Randbedingung für die Gruppierung von Kommunikationsobjekten innerhalb eines Abhängigkeitsgraphen soll sein, dass jede Gruppe komplett vorhanden sein muss oder komplett nicht vorhanden sein darf. Deswegen sollen die zwei Gruppen entweder disjunkt oder echte Teilmengen voneinander sein.

Es soll also für jede beliebige Paarung von zwei Gruppen von Kommunikationsobjekten A und B eine der drei folgenden Aussagen gelten:

1. Menge A und Menge B sind disjunkt. Das bedeutet, dass für jedes Element der Menge A gilt, dass es nicht Element der Menge B ist. Für jedes Element der

Menge B gilt, dass es nicht Element der Menge A ist. Mathematisch ausgedrückt wird dies durch folgende Aussage:

$$A \cap B = \emptyset :\Leftrightarrow \forall x \in A : x \notin B \wedge \forall x \in B : x \notin A \tag{7.36}$$

2. A ist echte Teilmenge von B. Das bedeutet, dass für jedes Element aus A gilt, dass es Element der Menge B ist. Außerdem gilt, dass A und B nicht gleich sind. In mathematischer Form wird dies wie folgt beschrieben:

$$A \subset B :\Leftrightarrow \forall x \in A : x \in B \wedge A \neq B \tag{7.37}$$

3. B ist echte Teilmenge von A. Das bedeutet, dass für jedes Element aus B gilt, dass es Element der Menge A ist. Außerdem gilt, dass A und B nicht gleich sind. In mathematischer Form wird dies wie folgt beschrieben:

$$B \subset A :\Leftrightarrow \forall x \in B : x \in A \wedge B \neq A \tag{7.38}$$

Um dies zu erreichen, müssen folgende Regeln bei der Gruppierung der Kommunikationsobjekte eingehalten werden:

1. In eine Gruppe dürfen nur Abhängigkeitspfeile hineingehen; es dürfen keine Abhängigkeitspfeile aus ihr herauskommen.

2. Eine neue Gruppe darf keine andere bestehende Gruppe schneiden und nicht echte Teilmenge von dieser sein.

3. Innerhalb einer Gruppe müssen alle Objekte eine Abhängigkeitsbeziehung zueinander haben, das heißt, dass alle Knoten innerhalb der Gruppe eine Verbindung aus Abhängigkeiten haben.

4. Schließen sich zwei Kommunikationsobjekte gegenseitig aus, so müssen sie ab einer gewissen Iteration des Gruppierungsvorganges in zwei unterschiedlichen Gruppen sein.

5. Alle verbindlich vorhandenen Kommunikationsobjekte sind in einer Gruppe zusammenzufassen.

Die Regeln 1 und 2 folgen direkt aus den Forderungen, die eingangs in diesem Absatz beschrieben wurden. Regel 3 hingegen stellt sicher, dass eine Gruppe nicht direkt aus mehreren unabhängigen Teilgruppen besteht.

Regel 4 dient sowohl der Verifikation des Gruppierungsvorganges und der Graphenerstellung als auch der Prüfung der zugrunde liegenden Spezifikation.

Durch Regel 5 wird eine Basisgruppe definiert, auf denen alle anderen aufbauen.

7.3.5 Ableitung von Profilen und deren Aufbau aus der Gruppierung von Kommunikationsobjekten

In Bild 7.11 zeigt ein Beispiel für einen gerichteten Graphen, der Abhängigkeiten einzelner Kommunikationsobjekte innerhalb einer beliebigen Spezifikation untereinander abbildet.

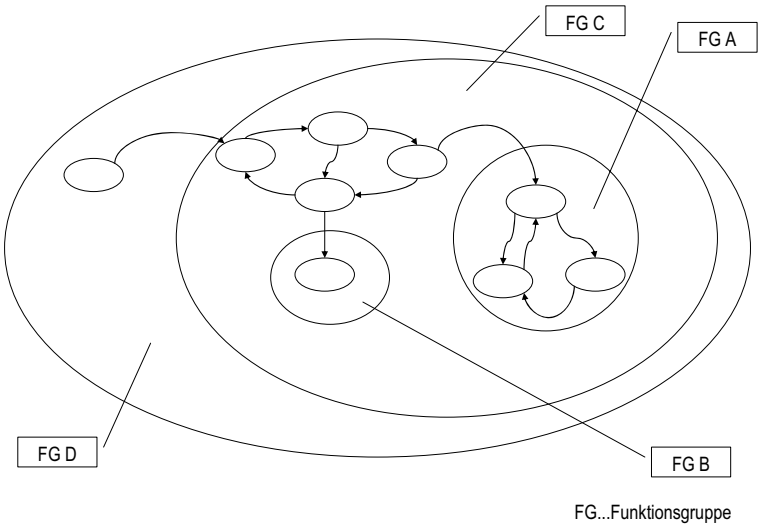


Bild 7.11: Beispiel für einen Abhängigkeitsgraphen

Die Kommunikationsobjekte innerhalb dieses Graphen wurden nach den in 7.3.4 definierten Regeln gruppiert. Dadurch ergeben sich vier Gruppen FG A, FG B, FG C und FG D.

Es gilt:

- FG A und FG B sind echte Teilmengen von FG C. ($A \subset C$, $B \subset C$)
- FG A, FG B und FG C sind echte Teilmenge von FG D. ($A \subset D$, $B \subset D$, $C \subset D$)
- FG A und FG B sind disjunkt. ($A \cap B = \emptyset$)

Es lassen sich mit Hilfe der Gruppierungen vereinfachte Abhängigkeitsbeziehungen zwischen den Kommunikationselementen ableiten:

- Aus der Differenzmenge von FG D und FG C folgt zwingend FG C. ($\mathrm{D} \setminus \mathrm{C} \Rightarrow \mathrm{C}$)
- Aus der Differenzmenge von FG C und der Vereinigungsmenge von FG A und FG B folgt zwingend FG A. ($\mathrm{C} \setminus (\mathrm{A} \cup \mathrm{B}) \Rightarrow \mathrm{A}$)
- Aus der Differenzmenge von FG C und der Vereinigungsmenge von FG A und FG B folgt zwingend FG B. ($\mathrm{C} \setminus (\mathrm{A} \cup \mathrm{B}) \Rightarrow \mathrm{B}$)
- Eine direkte Abhängigkeit zwischen A und B besteht nicht.

Diese Zusammenhänge können genutzt werden, um eine Profillandschaft innerhalb einer Spezifikation weiter zu strukturieren. Es sei im Beispiel die Menge FG D die Gesamtheit der Kommunikationsobjekte eines beliebigen Profils, welches weiter strukturiert werden soll.

Dann lassen sich innerhalb dieses Profils drei Ebenen ableiten, die unterschiedliche mögliche Implementierungsgrade darstellen. Diese Implementierungsgrade wären alle spezifikationskonform.

Das Ergebnis ist in Bild 7.12 dargestellt. FG A und FG B sind unabhängig voneinander und bilden die Basis für die zusätzlich zu FG A und FG B in FG C enthaltenen Kommunikationsobjekte. Erst wenn FG C komplett vorhanden ist, können die in FG D zusätzlich zu FG C enthaltenen Kommunikationsobjekte vorhanden sein.

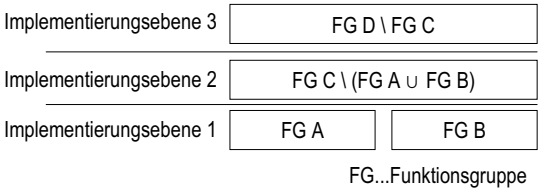


Bild 7.12: Implementierungsebenen innerhalb eines Profils

7.4 Interaktionsgraph von Kommunikationsobjekten

Über den endlichen gerichteten Multigraphen, wie in 7.3 beschrieben, hinaus ist auch ein weiterer Graph interessant. Der Graph aus 7.3.5 beschreibt nur die Abhängigkeiten bezüglich des Vorhandenseins von einzelnen Kommunikationsobjekten. Der weitere Graph soll Interaktionen von Kommunikationsobjekten untereinander darstellen. Das führt dazu, dass das Verhalten eines Gerätes bezüglich der Kommunikationsschnittstelle maschinenlesbar wird.

Ein solcher Graph dient einerseits in einem Engineeringwerkzeug dazu, um die Auswirkungen von Änderungen einzelner Kommunikationsobjekte vorherzusagen. Andererseits kann in einer Simulation das Verhalten eines virtuellen Gerätes beschreiben und so die Simulation bezüglich der Kommunikationsschnittstelle verbessert werden. Ist eine solche umfassende Beschreibung für ein spezifikationskonformes Gerät vorhanden, wird die Implementierung eines konkreten Gerätes sehr viel einfacher, da eine solche maschinenlesbare Spezifikation weniger Interpretationsfreiraum lässt, als es heutige Spezifikationen tun.

Mit einer solchen Beschreibung eines konformen Gerätes ist es auch für ein Testwerkzeug automatisiert vorhersehbar, wie sich ein konformes Gerät verhalten soll. Ein Konformitätstest kann damit weiter automatisiert werden, da viele Testfälle aus dem Graphen automatisiert ableitbar sind /86/.

Dieser Graph ist ungleich komplexer als der Graph aus 7.3.5, bietet aber auch die oben genannte Möglichkeit, das Verhalten eines Gerätes abzubilden und die damit verbundenen Vorzüge.

7.5 Klassifikation von Automatisierungsgeräten

Mit der in 7.4 hergeleiteten Methode wurden die Kommunikationsobjekte verschiedener Automatisierungsgeräte gruppiert. Im folgenden Abschnitt wird das Ergebnis dieser Arbeit dargestellt.

7.5.1 Funktionale Klassifikation von Automatisierungsgräten auf Kommunikationsebene

Auf der Ebene des Kommunikationsprofils finden sich unterschiedliche Gruppen von Funktionen. Die Ausprägung des Kommunikationsprofils hängt dabei stark von den Fähigkeiten des Kommunikationsmediums ab. So können einige Funktionsgruppen bei bestimmten Medien nicht vorhanden sein, weil das Medium die dazu notwendigen Fähigkeiten nicht bietet.

Folgende Funktionsgruppen existieren auf Ebene des Kommunikationsprofils:

- Buskonfiguration
- Kommunikationskonfiguration
- Verbindungskonfiguration
- Busdiagnose
- Synchronisation
- Überwachung
- Redundanz
- Hotplug
- Echtzeitbits
- Übertragungsmechanismen
- Nichtechtzeitkommunikation
- Bedarfsdaten

In der Funktionsgruppe **Buskonfiguration** sind alle Funktionen zusammengefasst, die zur Konfiguration des Busses notwendig sind. Sie beinhaltet Funktionen wie die Arbitrierung des Busses.

Die Funktionsgruppe **Kommunikationskonfiguration** beinhaltet alle Funktionen die zur Konfiguration der grundlegenden Kommunikationseigenschaften der Busteilnehmer notwendig sind. Dies beinhaltet beispielsweise die Konfiguration des Bustaktes.

In der Funktionsgruppe **Verbindungskonfiguration** werden die Funktionen zusammengefasst, die zur Konfigurierung der Busteilnehmer auf Verbindungsebene benötigt werden, wie zum Beispiel die Konfigurierung der Verbindungsteilnehmer, deren Rolle in der Verbindung, die Art der Verbindung und die Zykluszeit.

Die Funktionsgruppe **Busdiagnose** vereint die Funktionen, die benötigt werden, um den Zustand des Busses zu überwachen und Fehler zu erkennen.

Alle Funktionen, die zum synchronisierten Betrieb von Busteilnehmern und taktsynchronen Verbindungen benötigt werden, sind in der Funktionsgruppe **Synchronisation** zusammengefasst.

Funktionen zur Überwachung von Verbindungen, die nicht taktsynchron, sondern mittels eines Watchdogs überwacht werden, finden sich in der Funktionsgruppe **Überwachung**.

Wird durch das Bussystem eine hardwareseitige Redundanz gegen das Versagen einzelner Komponenten wie Kabel oder auch ganzer Teilnehmer angeboten, so finden sich alle Funktionen zum Abfragen des Status und der Steuerung der Ausfallsicherheit in der Funktionsgruppe **Redundanz**.

Zur echtzeitfähigen Steuerung von Funktionen und zur echtzeitfähigen Abfrage von Diagnosen und Statusmeldungen von Automatisierungsgeräten werden sogenannte Echtzeitbits verwendet. Die zur Konfiguration dieser Bits notwendigen Funktionen werden in der Funktionsgruppe **Echtzeitbits** zusammengefasst.

Bietet das Bussystem transparente Funktionen zum Multiplexen und zur Fragmentierung von Daten oder ganzen Verbindungen, so finden sich diese in der Funktionsgruppe **Übertragungsmechanismen**.

Die Funktionsgruppe **Nichtechtzeitkommunikation** gruppiert alle Funktionen, die dazu verwendet werden, nichtechtzeitfähige Protokolle, beispielsweise aus der Bürokommunikation, auf dem Bussystem zu unterstützen.

Die zum Zugriff und zur Manipulation von Bedarfsdaten notwendigen Funktionen werden in der Funktionsgruppe **Bedarfsdaten** zusammengefasst.

7.5.2 Funktionale Klassifikation von Automatisierungsgeräten auf Applikationsebene

Wie bereits in 7.1.2 beschrieben, fanden sich Kommunikationsobjekte auf der Applikationsebene, die nicht eindeutig einer Klasse von Geräten zuordenbar sind, sondern bei mehreren Klassen vorhanden sind.

Durch diese Kommunikationsobjekte werden Funktionen abgedeckt, die sich in folgenden Funktionsgruppen einordnen lassen:

- Identifikation
- Diagnose
- Administration
- Archivierung
- Zustandsmaschine

Die Funktionsgruppe **Identifikation** beinhaltet alle Kommunikationsobjekte, die dazu dienen, das Gerät zu erkennen. Hierzu dienen typischerweise sogenannte elektronische Typenschilder, die mindestens eine eindeutige Hersteller- und Produktkennung bereitstellen.

In der Funktionsgruppe **Diagnose** sind alle Kommunikationsobjekte zusammengefasst, die dazu dienen, Zustände des Gerätes zu erkennen und darauf zuzugreifen. Es wird

hierbei zwischen Fehlern, Warnungen und Informationen unterschieden, Ebenfalls enthalten sind Kommunikationsobjekte zum Zurücksetzen dieser Diagnosen.

Die Funktionsgruppe **Administration** fasst alle Kommunikationsobjekte zusammen, die mit administrativen Aufgaben im Gerät, wie der Sprachauswahl und dem Passwortschutz einzelner Kommunikationsobjekte verbunden sind,

In der Funktionsgruppe **Archivierung** sind alle Kommunikationsobjekte beinhaltet, die für Aufgaben zur Sicherung und Wiederherstellung von Gerätekonfigurationen und Geräteparametrierungen benötigt werden.

Die Funktionsgruppe **Zustandsmaschine** fasst alle Kommunikationsobjekte zusammen, die zur Steuerung und Anzeige der Gerätezustandsmaschine notwendig sind.

7.5.2.1 Funktionale Klassifikation von modularen E/A-Geräten

Modulare E/A-Geräte sollen die Vielfalt der Signale aus den Feldanschaltungen konzentrieren und an eine übergeordnete Steuerungseinheit wie eine SPS weitergeben. Neben der Verarbeitung der zusammengefassten Signale durch die SPS müssen einzelne Signale auch als Querverbindungen für andere Busteilnehmer aufgeschaltet werden können. Moderne E/A-Systeme integrieren neben einer großen Zahl von verschiedenen digitalen und analogen Funktionen auch Kommunikationsmodule, Gateways zu Subbussen und Sicherheitsmodule /87/, /88/, /89/. Die Module innerhalb einer solchen Station werden über einen Lokalbus verbunden und von einem sogenannten Buskopf gesteuert, der zum überlagerten Feld- oder Antriebsbus hin als Slave in Erscheinung tritt.

Die flexible Anordnung der E/A-Module bedingt, dass der Buskopf nach einem Neustart eine Erkennung der E/As auf dem Lokalbus vornimmt und die sich ergebenden Funktionen und Prozessdaten in Informationsstrukturen an den überlagerten Feld- oder Antriebsbus weitergibt.

Digitale E/As weisen eine Bitgranularität auf - jedes Bit entspricht also einem Schaltzustand des Ein- oder Ausgangs. Analoge Module werten typischerweise 12 Bit-, 16 Bit- oder 24 Bit-Prozessdaten aus, was eine Codierung der Prozessdaten als 2 oder drei Bytes ergibt. Historisch bedingt werden in den SPS-Programmen die Informationen als Bytes, Worte oder Doppelworte behandelt.

Innerhalb dieser Klassifikation sind Mehrfachvererbungen vorgesehen. Das bedeutet, dass ein Modul mehren Klassen angehören kann. Somit ist eine funktionale Zuordnung auch von Modulen möglich, die beispielsweise sowohl digitale Ein- aus auch Ausgänge haben.

Die wichtigsten Parameter innerhalb der Funktionsgruppen befassen sich mit den Prozessdaten: So können vom Master Kanalzahl und Datenbreite, sowie der Dateninhalt für den Eingang und den Ausgang erfragt werden. Ebenso kann mit Diagnosedaten und deren Quittierung verfahren werden, für sie sind ebenfalls die Kanalzahl und die Datenbreite, wie auch der Inhalt hinterlegt. Eine herstellerspezifische Parametrierung kann für den Fall der Erstparametrierung, wie auch des Neuanlaufes oder des Modultausches über den sog. Parameterkanal erreicht werden. Die Module geben über den Buskopf Auskunft über ihr Timingverhalten vom Eingang über den Lokalbus zum überlagerten Feld- oder Antriebsbus.

7.5.2.2 Funktionale Klassifikation von Antriebsgeräten

Die Regelung von Antriebsgeräten folgt grundsätzlich der in Bild 7.13 gezeigten Struktur /90/. Die Aufteilung der Funktionen zwischen der Bewegungssteuerung und dem Antriebsgerät kann dabei stark variieren. Je nachdem welche der beschriebenen Schnittstellen vom Antrieb implementiert sind und nach außen als Kommunikationsobjekt zur Verfügung gestellt wird, ergeben sich vier Klassen von elektrischen Antriebsgeräten:

- Antriebsgeräte mit Momentenschnittstelle,
- Antriebsgeräte mit Drehzahlschnittstelle,
- Antriebsgeräte mit Lageschnittstelle,
- Antriebsgeräte mit Positionierschnittstelle.

Ein Antriebsgerät kann aber auch mehrere dieser Schnittstellen anbieten und je nach Parametrierung die eine oder andere im Betrieb bedienen und sogar zwischen unterschiedlichen Schnittstellen umgeschaltet werden.

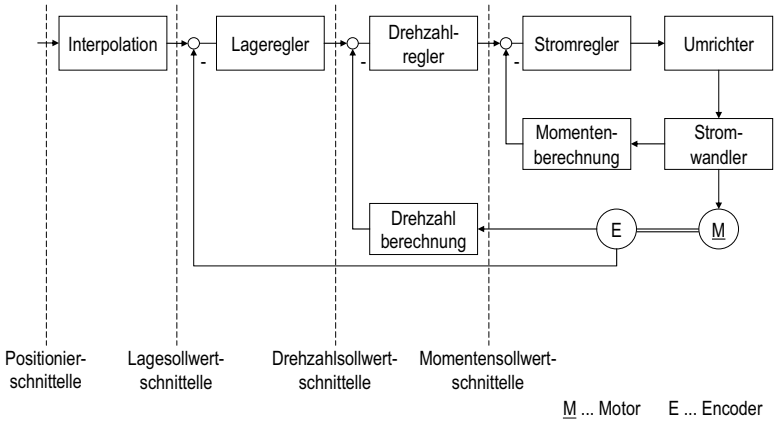


Bild 7.13: Struktur eines Antriebsgerätes

7.6 Zusammenfassung

In diesem Kapitel wurden zunächst die Notwendigkeit und die Anwendung von Profilen zur funktionalen Klassifikation von Automatisierungsgeräten hergeleitet. Hierzu wurden zwei Metaprofilebenen, das Kommunikationsmetaprofil und Applikationsmetaprofil, basierend auf dem OSI-Referenzmodell abgeleitet. Es zeigte sich, dass auf Applikationsebene eine weitere Unterteilung in generische und funktionsspezifische Kommunikationsobjekte notwendig ist.

In einem weiteren Schritt wurde eine Methode hergeleitet, die es erlaubt, Kommunikationsobjekte zu Profilen und Funktionsgruppen zusammenzufassen. Dies geschieht durch die Auswertung von Abhängigkeiten der Objekte bezüglich des Vorhandenseins untereinander. Die Methode nutzt dabei Verfahren aus der Mengen- und Graphentheorie.

Die Methode wurde dann nochmals erweitert. So bietet sie die auch die Möglichkeit, unter Einbeziehung des gegenseitigen Ausschlusses von Kommunikationsobjekten Fehler bei der Erstellung des abgeleiteten Graphen oder sogar der Spezifikation zu erkennen. Hierzu wird ein weiterer Graph abgeleitet und dem Abhängigkeitsgraphen

überlagert. Durch Bilden der transitiven Hülle des einen und Vergleich mit dem anderen Graphen können potentielle Fehler leicht erkannt werden.

Die Methode wurde mathematisch beschrieben und ist automatisierbar. Das heißt, es ist möglich, sie in einem Programm umzusetzen.

Des Weiteren wurden die Funktionen, die durch die abgeleiteten Profilebenen abgedeckt werden, beschrieben. Hierbei wurden die Funktionen sowohl aus der Kommunikation als auch auf Applikationsebene in Funktionsgruppen eingeteilt.

8 Beschreibungsmöglichkeit für die funktionale Typisierung von Automatisierungsgeräten im Informationsmodell

In diesem Kapitel werden zunächst die Informationsmodelle für die Beschreibung von Profilen und Geräten als Klassendiagramme hergeleitet. Für diesen Schritt werden Methoden zur Erstellung formaler Sprachen aus der Informatik verwendet. Zuletzt werden die beiden Informationsmodelle zusammengeführt, um eine Beschreibungsmöglichkeit für funktional typisierte Automatisierungsgeräte zu erhalten.

8.1 Informationsmodell für die Darstellung von Profilen

Ziel ist es, eine Darstellung für Profile zu finden, die den in den vorherigen Kapiteln abgeleiteten Anforderungen genügt. Hierbei sollen vor allem die Definitionen von Kommunikationsobjekten, Profilen, Subprofilen und Funktionsgruppen, wie sie in Kapitel 2 eingeführt wurden, berücksichtigt werden. Hinzu kommt die Notwendigkeit, die Abhängigkeiten und Interaktionen von Kommunikationsobjekten abzubilden.

8.1.1 Klassendarstellung des Informationsmodells für die Darstellung von Profilen

Zunächst wird das Informationsmodell für die Darstellung von Profilen als Klassendiagramm mittels der Spezifikationssprache UML abgeleitet.

8.1.1.1 Die Klasse Profile

Die Basis der Beschreibung eines Profils bildet die Klasse „Profil“ wie sie in Bild 8.1 dargestellt ist.

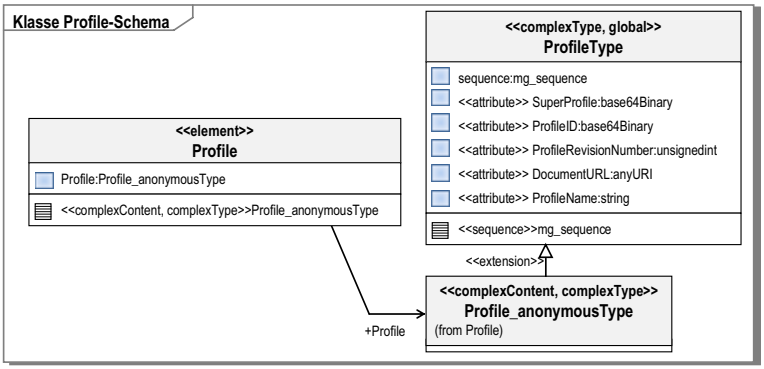


Bild 8.1: Klassendiagramm der Klasse „Profile

Ein Profil besteht aus mehreren sogenannten Profilelementen. Ein Profil selber besitzt einen Profilnamen, eine Kennzeichnungsnummer zur Identifikation und eine Revisionsnummer, um eine Versionierung des Profils zu erlauben. Hinzu kommt optional eine Referenz auf ein Superprofil. Dies dient dazu, um Hierarchien von Profilen abzubilden.

8.1.1.2 Die Klasse ProfileElement

Ein Profilelement wird durch die Klasse „ProfileElement“ dargestellt und besteht aus einem oder mehreren Funktionsgruppen, wie es in Bild 8.2 dargestellt ist. Ein solches Profilelement besitzt wiederum einem Namen, eine Beschreibung in Klartext und die Eigenschaft, ob es verpflichtend in dem beschriebenen Profil vorhanden ist.

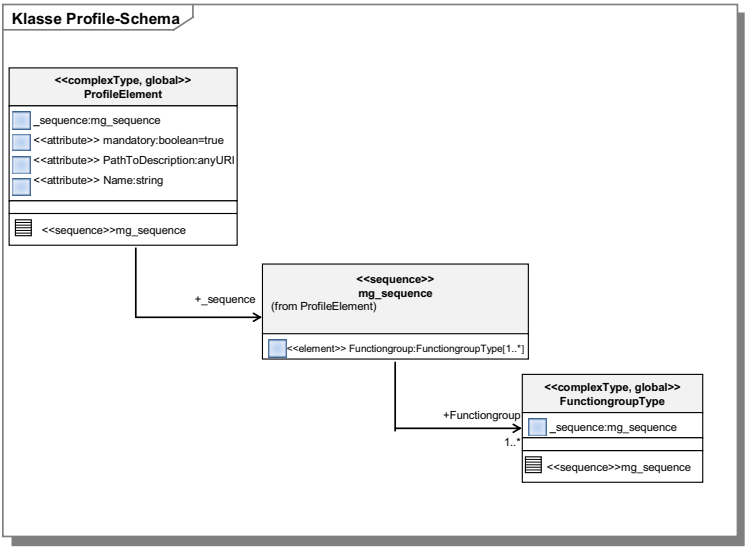


Bild 8.2: Klassendiagramm der Klasse „ProfileElement“

8.1.1.3 Die Klasse FunctionGroupType

Eine Funktionsgruppe wird durch die Klasse „FunctionGroupType“ repräsentiert und besteht, wie in Bild 8.3 dargestellt, aus einem oder mehreren Kommunikationsobjekten. Diese sind innerhalb der Funktionsgruppe entweder verpflichtend vorhanden oder nicht.

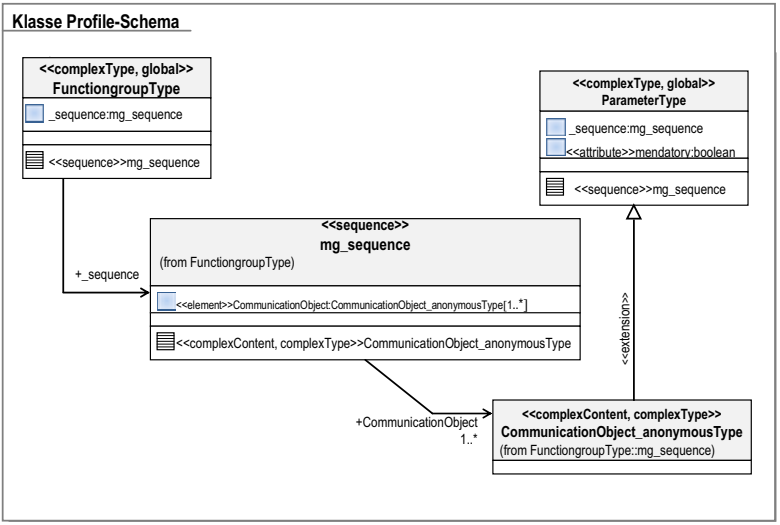


Bild 8.3: Klassendiagramm der Klasse „FunctionGroupType“

8.1.1.4 Die Klasse ParameterType

Ein Kommunikationsobjekt wird durch die Klasse „ParameterType“ dargestellt und besteht aus Daten und hat gegebenenfalls Eigenschaften wie in Bild 8.4 gezeigt. Die Möglichkeit des Objektes, dass es verpflichtend vorhanden sein muss, gibt die Möglichkeit, einem Kommunikationsobjekt die in 7.3.2.2 hergeleiteten Beziehungen abzubilden.

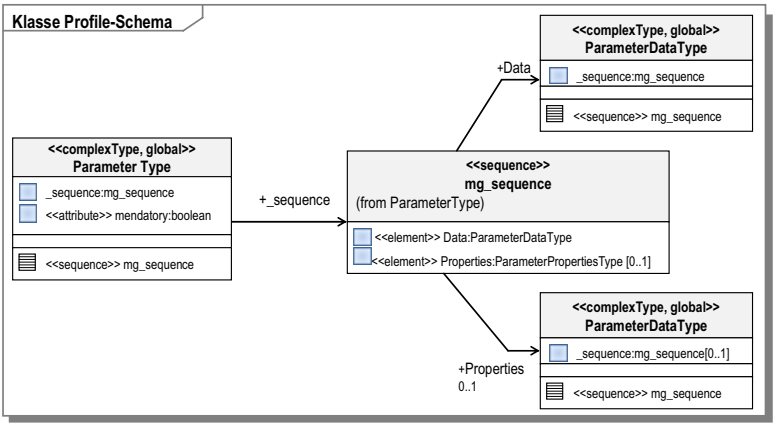


Bild 8.4: Klassendiagramm der Klasse „ParameterType“

8.1.1.5 Die Klasse ParameterDataType

Die Daten eines Kommunikationsobjektes werden durch die Klasse „ParameterDataType“ repräsentiert und lassen sich wie in Bild 8.5 dargestellt zusammenfassen. Das Objekt hat einen Namen und einen Identifier, um es ansprechen zu können.

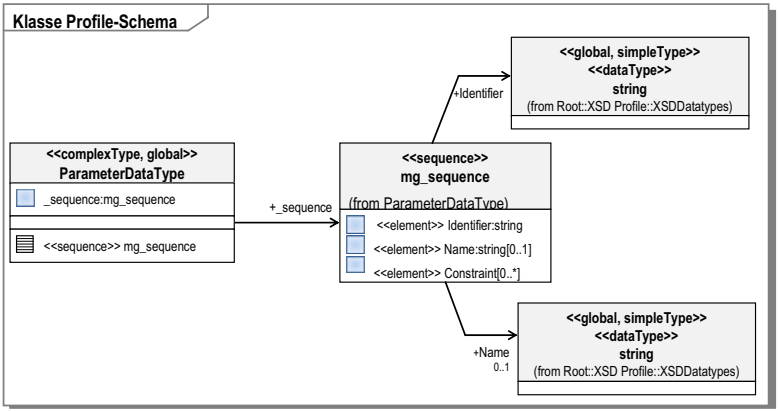


Bild 8.5: Klassendiagramm der Klasse „ParameterDataType“

8.1.1.6 Die Klasse ParameterPropertiesType

Die Eigenschaften eines Kommunikationsobjektes werden durch die Klasse „ParameterPropertiesType“ und sind in Bild 8.6 gezeigt. Ein Kommunikationsobjekt kann zyklisch ansprechbar sein und es kann schreibbar sein, was sich in der Klasse „WriteableType“ (siehe Kapitel 8.1.1.7) widerspiegelt.

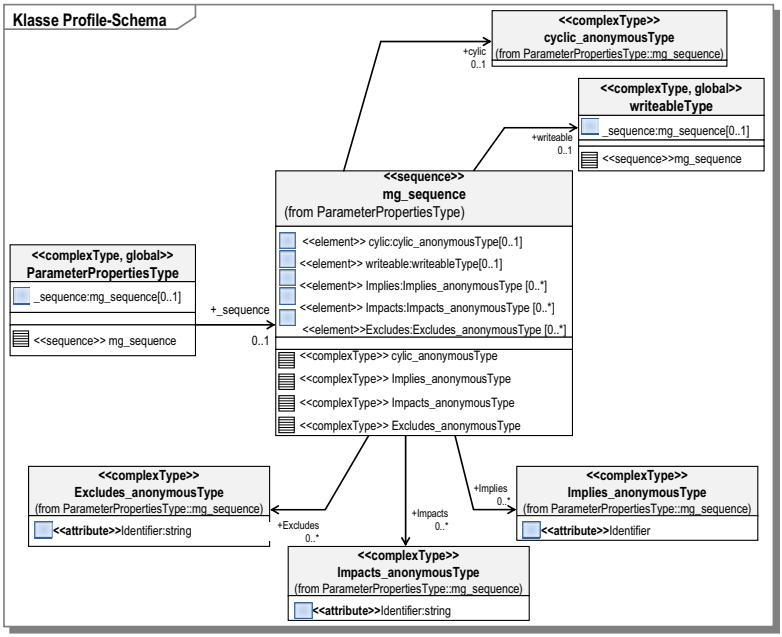


Bild 8.6: Klassendiagramm der Klasse „ParameterPropertiesType“

Der Abhängigkeitsgraph aus 7.3.2.1 wird dadurch in dem Modell darstellbar, dass hier eine Beziehung zu anderen Kommunikationsobjekten mittels einer Beziehung mit der Semantik Impliziert (Impacts in Bild 8.6) dargestellt werden kann. Der Graph aus 7.3.2.3, der den gegenseitigen Ausschluss von Kommunikationsobjekten darstellt, kann mittels der Beziehung mit der Semantik Schließt Aus (in Bild 8.6 „Excludes“) erstellt werden.

Der Interaktionsgraph, der in 7.4 beschrieben wurde, kann mittels der Beziehung mit der Semantik Hat Einfluss (in Bild 8.6 „Impacts“) abgebildet werden.

8.1.1.7 Die Klasse WriteableType

Die Eigenschaft des Schreibbarsein eines Kommunikationsobjektes wird durch die Klasse „WritableType“ in Bild 8.7 abgebildet. Ein Kommunikationsobjekt kann in

unterschiedlichen Phasen der Kommunikation und des Betriebs die Eigenschaft des Schreibbarseins ändern.

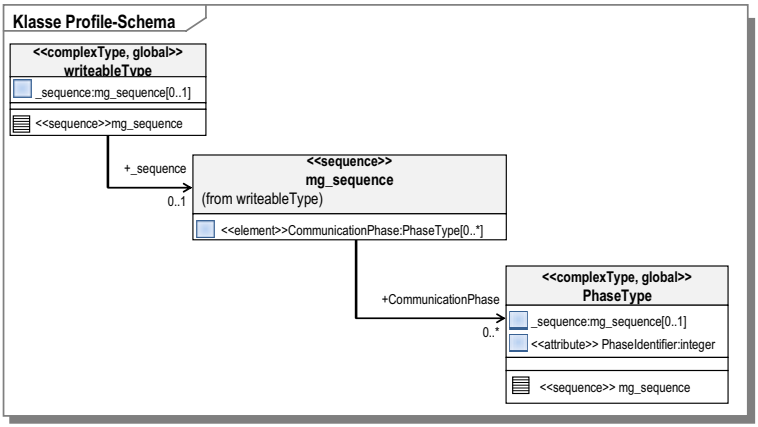


Bild 8.7: Klassendiagramm der Klasse „WriteableType“

8.1.1.8 Die Klasse PhaseType

Wird in ein Kommunikationsobjekt geschrieben, kann es zu Fehlern kommen, die in den entsprechenden Phasen der Kommunikation und des Betriebs unterschiedlich sind. Dies wird durch die Klasse „PhaseType“ repräsentiert und ist in Bild 8.8 dargestellt. Jeder Kommunikationsphase können hierbei unterschiedliche Fehler zugeordnet werden.

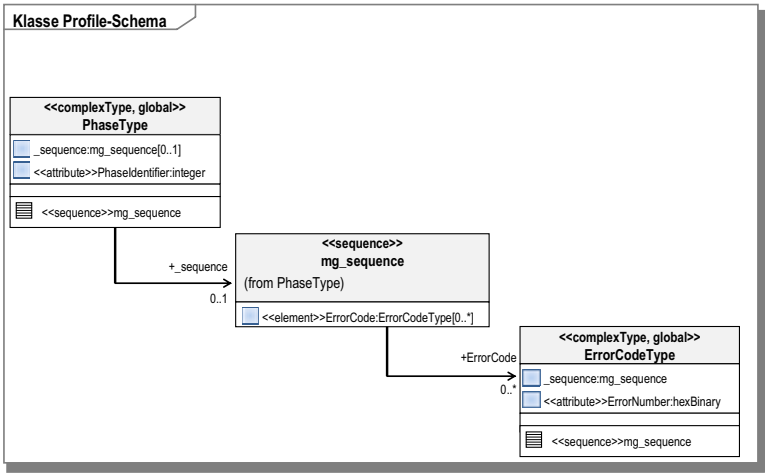


Bild 8.8: Klassendiagramm der Klasse „PhaseType“

8.1.1.9 Die Klasse ErrorcodeType

Ein Fehler wird durch eine Fehlernummer gekennzeichnet. Dieser Fehlernummer muss sprachenunabhängig eine Klartextanzeige zuordenbar sein. Um dieses Problem zu lösen wurde es möglich gemacht, jeder Fehlernummer eine sogenannte Internationalisierungstabelle zur Klartextdarstellung in unterschiedlichen Sprachen bereitzustellen. Dieser Zusammenhang wird in der Klasse „ErrorcodeType“ dargestellt, wie es in Bild 8.9 gezeigt ist.

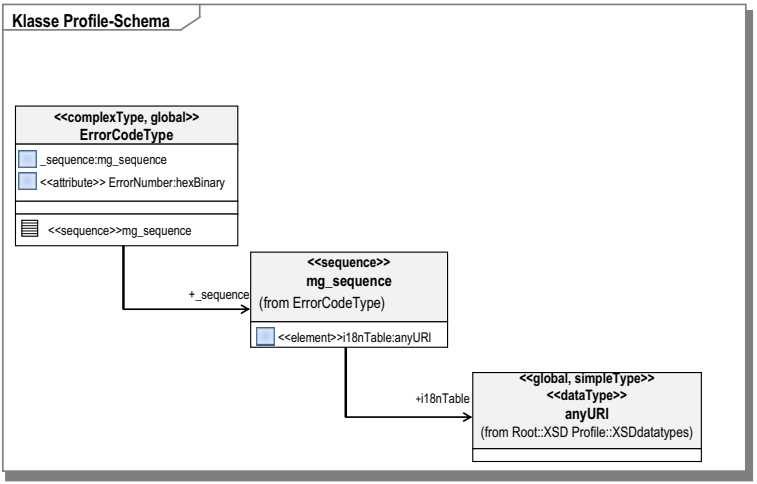


Bild 8.9: Klassendiagramm der Klasse „ErrorcodeType“

8.1.2 Umsetzung in einem XML-Schema

Das in 8.1.1 abgeleitete Modell wurde durch Transformation aus der UML-Darstellung in ein XML-Schema umgewandelt. Dieses Schema dient als Grundlage für die textuelle und maschinenlesbare Form der Beschreibung von Profilen, deren Aufbau und Abhängigkeiten.

Die Kommunikationsobjekte innerhalb von Profilen können damit in ihren Eigenschaften sowohl maschinen- als auch menschenlesbar dargestellt werden. Es ist möglich, aus der Darstellung die in 7.4 eingeführten Graphen zu generieren und die entsprechenden Schlüsse bezüglich Gruppierbarkeit und Validität des Modells zu ziehen.

8.2 Informationsmodell zur Darstellung von Geräten

Ziel des Informationsmodells zur Darstellung von Geräten ist es, eine funktionale Sicht auf Automatisierungsgeräte zu bieten. Diese funktionale Sicht soll dabei den in Kapitel 6 abgeleiteten Anforderungen genügen, das heißt, dass eine Klassifizierung der Geräte auf

drei Ebenen erfolgt. Diese Ebenen sind die Kommunikationsebene und die zwei abgeleiteten Applikationsebenen.

8.2.1 Klassendarstellung des Informationsmodells für die Darstellung von Geräten

Zunächst wird das Informationsmodell in Form von UML-Diagrammen beschrieben.

8.2.1.1 Die Klasse DeviceType

Die Klasse „DeviceType“ stellt das Gerät dar. Ein Gerät hat aus funktionaler Sicht mehrere Kommunikationsschnittstellen, über die es Funktionen an andere Teilnehmer bereitstellt, wie es in Bild 8.10 gezeigt ist.

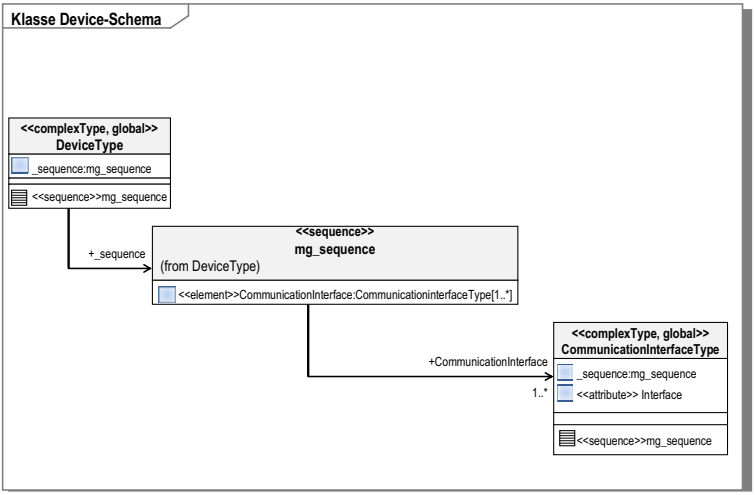


Bild 8.10: Klassendiagramm der Klasse „DeviceType“

Es können an einem Gerät mehrere Kommunikationsschnittstellen vorhanden sein. Sie werden jeweils durch einen Typ gekennzeichnet.

8.2.1.2 Die Klasse CommunicationInterfaceType

Die Kommunikationsschnittstellen eines Gerätes werden durch die Klasse „CommunicationInterfaceType“ dargestellt. Auf jeder Kommunikationsschnittstelle können ein oder mehrere Kommunikationsprotokolle durch das Gerät bedient werden, wie es in Bild 8.11 gezeigt ist. So ist es beispielsweise möglich auf einer RJ45-Schnittstelle nicht nur Standardethernet sondern auch eine Echtzeitethernet-Variante gleichzeitig zu bedienen.

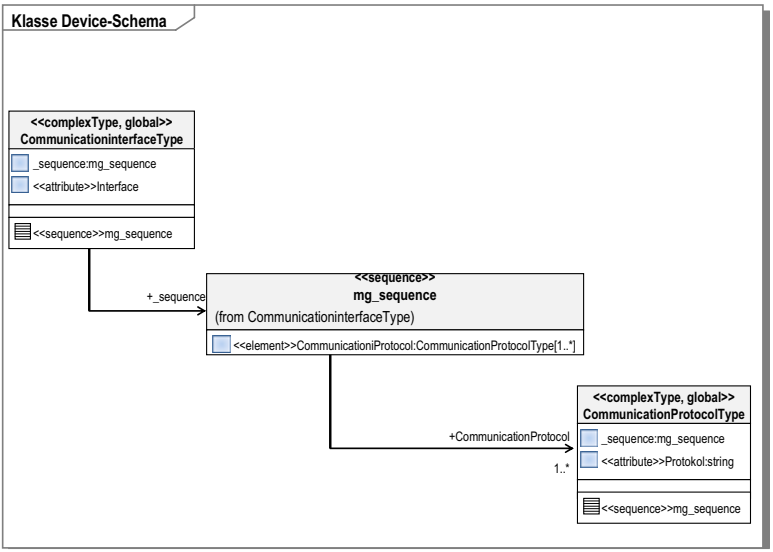


Bild 8.11: Klassendiagramm der Klasse „CommunicationInterfaceType“

8.2.1.3 Die Klasse CommunicationProtocolType

Die Klasse „CommunicationProtocolType“ ist in Bild 8.12 dargestellt, sie stellt die Kommunikationsprotokolle dar, die an einer Schnittstelle verfügbar sind. Aus Sicht eines Kommunikationsprotokolls besitzt das Gerät einen bestimmten Typ, der im Gerätetyp bereitgestellt wird. Die kommunikativen Funktionen, die das Gerät aus dem Kommunikationsprotokoll unterstützt, werden durch das Kommunikationsprofil beschrieben. Ein solches Kommunikationsprofil lässt sich über eine Profil- und eine

Funktionen, die das Gerät aus dem Kommunikationsprotokoll unterstützt, werden durch das Kommunikationsprofil beschrieben. Ein solches Kommunikationsprofil lässt sich über eine Profil- und eine Revisionsnummer beschreiben. Auf diese Weise ist es möglich, eine Referenz auf eine externe Profilbeschreibung zu geben.

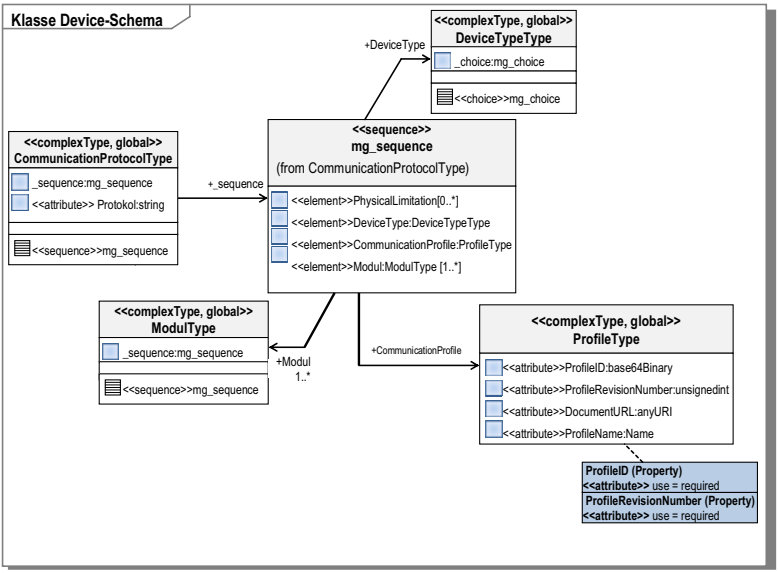


Bild 8.12: Klassendiagramm der Klasse „CommunicationProtocolType“

Des Weiteren besteht ein Gerät aus einem oder mehreren Modulen, wie es in 6.2.3 abgeleitet wurde. Diese modulare Sicht ist in einer weiteren Klasse beschrieben.

8.2.1.4 Die Klasse DeviceTypeType

Die Eigenschaft des Gerätes, ob es in einem Kommunikationsprotokoll als Master oder als Slave fungiert, wird in der Klasse „DeviceTypeType“ beschrieben. Diese Klasse ist in Bild 8.13 dargestellt.

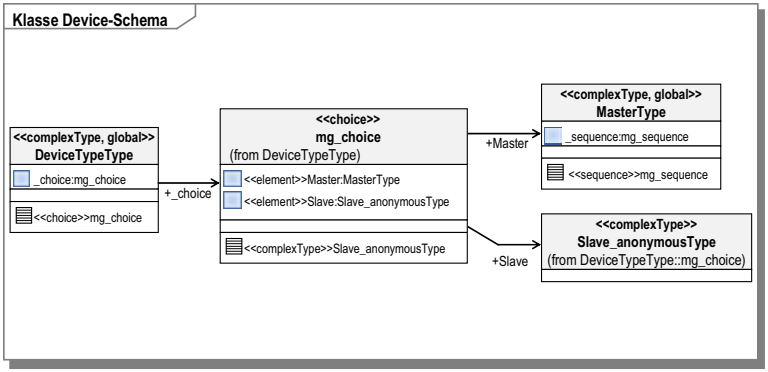


Bild 8.13: Klassendiagramm der Klasse „DeviceTypeType“

8.2.1.5 Die Klasse MasterType

Handelt es sich bei einem Gerät um einen Master, so kann an den Master eine Applikationsbeschreibung in Form einer Liste von benötigten Profilen in einem bestimmten Slave im Kommunikationsverbund angegeben werden. Dieser Umstand wird in der Klasse „MasterType“ dargestellt, das Klassendiagramm der Klasse „MasterType“ ist in Bild 8.14 gezeigt.

Diese Applikationsbeschreibung dient dazu, in einem Engineeringwerkzeug testen zu können, ob die verwendeten Geräte die durch die Applikation benötigten Funktionen bereitstellen.

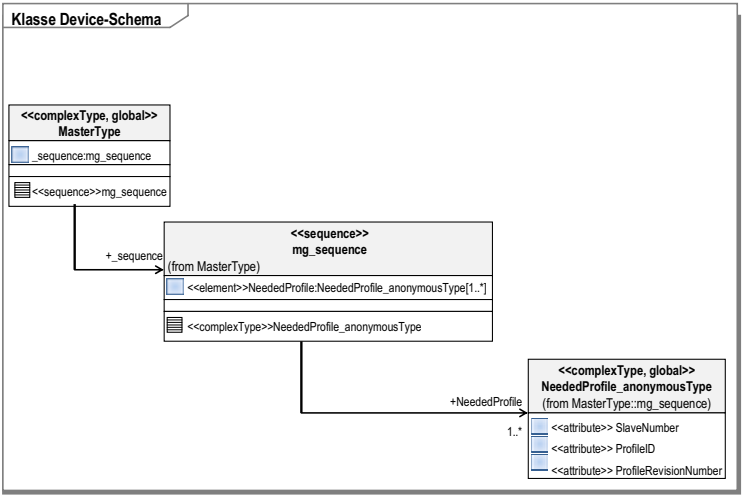


Bild 8.14: Klassendiagramm der Klasse „MasterType“

8.2.1.6 Die Klasse „ModulType“

Die Klasse „ModulType“ ist in Bild 8.15 gezeigt. Sie dient dazu, das für ein Modul bereitgestellte Applikationsprofil durch eine Referenz zugänglich zu machen.

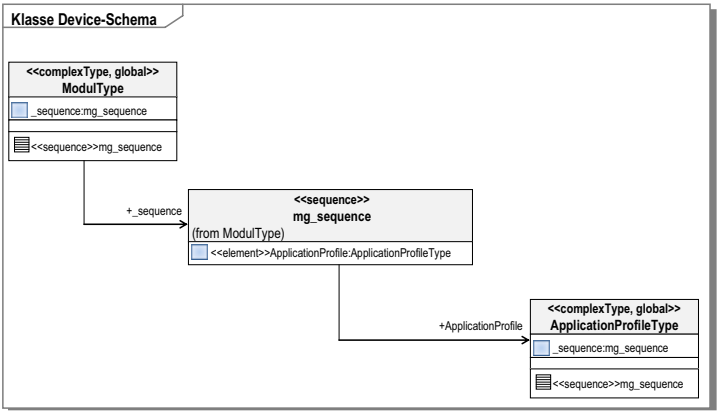


Bild 8.15: Klassendiagramm der Klasse „ModuleType“

8.2.1.7 Die Klasse ApplicationProfileType

Ein Applikationsprofil besteht aus einem generischen Teil und einem funktionsspezifischen Teil wie es in 7.1.2 beschrieben wurde. Diese beiden Arten lassen sich jeweils wieder durch eine Profil- und eine Revisionsnummer eindeutig zuordnen. Dieser Umstand wird in der Klasse „ApplicationProfileType“ dargestellt, wie in Bild 8.16 gezeigt ist.

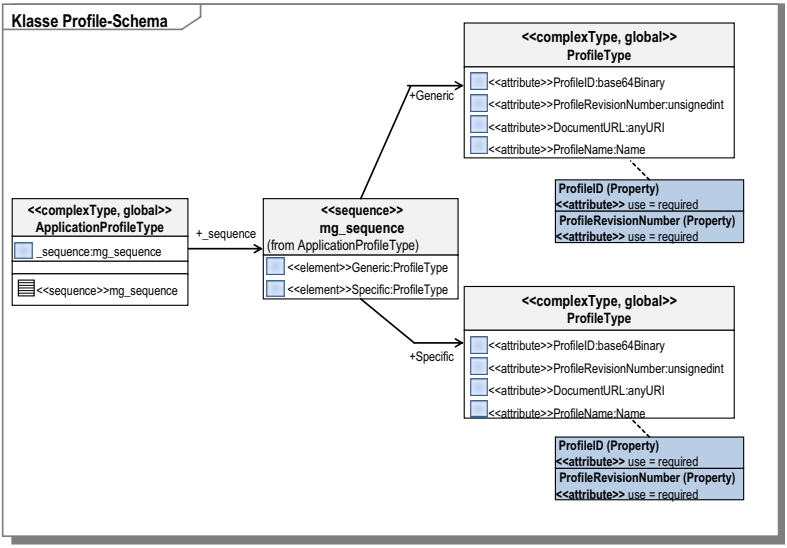


Bild 8.16: Klassendiagramm der Klasse „ApplicationProfileType“

8.2.2 XML-Schema des Informationsmodells für die Darstellung von Geräten

Wie schon bei dem Informationsmodell für die Darstellung von Profilen lässt sich auch das Informationsmodell für die Darstellung von Geräten aus der Darstellung in UML in ein XML-Schema übertragen.

Ergebnis ist eine Beschreibungsform für Geräte, die es erlaubt, den funktionalen Aufbau von Automatisierungsgeräten darzustellen. Dies geschieht durch Referenzen auf

Profilbeschreibungen. Diese Profilbeschreibungen können in Form des in 8.1 abgeleiteten Informationsmodells dargestellt sein.

8.3 Zusammenfassung

In diesem Kapitel wurden Informationsmodelle abgeleitet, die die Beschreibung von Profilen einerseits und die funktionale Beschreibung von Automatisierungsgeräten andererseits umfassen.

Bei der Umsetzung wurden die in vorherigen Kapiteln abgeleiteten Anforderungen berücksichtigt. So besitzt die Profilbeschreibung die Möglichkeit, die Information der in Kapitel 7 abgeleiteten Graphen zu transportieren. Die Beschreibung der Automatisierungsgeräte stützt sich auf die Profilbeschreibungen und bietet so eine funktionale Sicht auf die Geräte. Hinzu kommt noch die Möglichkeit, nicht nur die bereitgestellten Funktionen von Geräten zu beschreiben, sondern auch die Anforderungen, die eine Applikation an Funktionen benötigt, damit sie umgesetzt werden kann.

9 Realisierung

Die Methoden und Ergebnisse, die in den vorangehenden Kapiteln beschrieben worden sind, wurden im Rahmen von Forschungsprojekten am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen (ISW) an der Universität Stuttgart entwickelt und flossen durch die Mitarbeit in Normungsgremien in die Gremienarbeit ein /91/, /92/, /93/, /94/, /95/, /96/, /97/, /98/, /99/, /100/, /101/.

9.1 Methode zum Vergleich der Mächtigkeit von Beschreibungssprachen

Die in Kapitel 5 hergeleitete Methode zum Vergleich der Mächtigkeit von Beschreibungssprachen wurde in zwei Forschungsprojekten verwendet, um Gerätebeschreibungssprachen und die zu Grunde liegenden Gerätemodelle zu vergleichen.

Ziel der Projekte war jeweils, die Objekte innerhalb der jeweiligen Gerätemodelle zu identifizieren, die im Ausgangs- oder Zielmodell nicht vorhanden sind. Auf Grund dieser Erkenntnisse wurden Konverter entworfen, die es erlaubten, mittels einer zugeschalteten Datenbasis eine Konvertierung zu implementieren und diese mit zusätzlich notwendigen Daten anzureichern.

Um Gerätebeschreibungsdateien eines Ausgangsformats in ein Zielformat zu transformieren, wurde das in Bild 9.1 gezeigte Prinzip verwendet. Grundsätzlich wird das Ausgangsformat durch Transformation in das Zielformat verwandelt. Da die meisten aktuellen Gerätebeschreibungssprachen XML als Darstellungsform verwenden, bietet sich hier eine Transformation mittels XSLT (Extensible Stylesheet Language Transformation) an. Diese transformiert die vorhandenen Elemente der Ausgangssprache in die Elemente der Zielsprache. Dies kann entweder durch einfache Beziehungen wie Umkopieren oder durch komplexe Beziehungen wie das Auswerten mehrerer Elemente zur Erstellung eines Elementes geschehen.

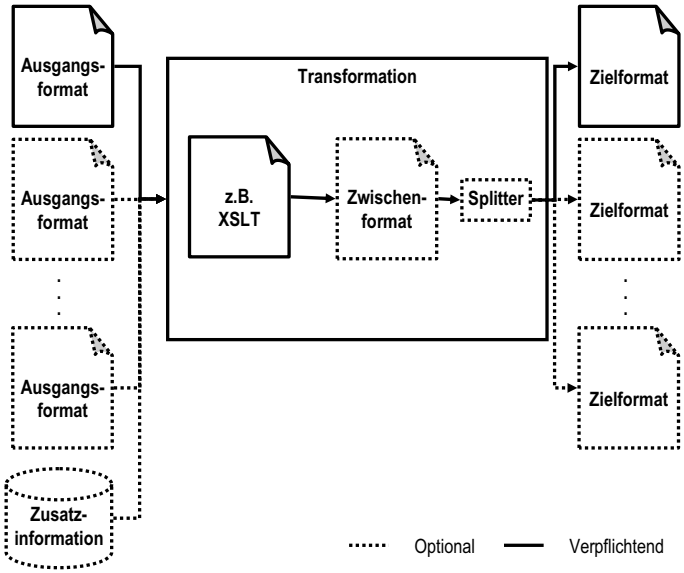


Bild 9.1: Transformation von Gerätebeschreibungssprachen

Wurde beim Vergleich der zugrunde liegenden Modelle herausgefunden, dass notwendige Informationen zur Transformation im Ausgangsmodell nicht vorhanden sind, so können diese in Form von Zusatzinformationen in einer optionalen Datenbasis bereitgestellt werden, falls sie für das Ergebnis notwendig oder gewünscht sind.

Unterschiedliche Beschreibungssprachen ziehen die Dateigrenzen auf verschiedene Arten. So gibt es Sprachen, die alle möglichen Module eines modularen Gerätes in einer Datei unterbringen (zum Beispiel GSDML), andere benutzen für jedes Modul eine eigene (zum Beispiel FDCML). Dieser Tatsache sind die anderen optionalen Teile in der Transformationskette geschuldet. Einerseits können am Eingang mehrere Dateien stehen, andererseits kann ein optionales Zwischenformat, welches durch einen Splitter in mehrere Ausgangsdateien gewandelt werden kann. Diese Tatsache hat nichts mit dem zugrunde liegenden Modell zu tun, sondern ist ausschließlich eine Sache der Umsetzung des Modells in der verwendeten Darstellungsform.

9.2 Strukturierung von Profilen

Als Beispiel für die Umsetzung der Strukturierung eines Profils in Funktionsgruppen und deren Zuordnung zu den Klassen von Antriebsgeräten soll das Antriebsprofil des SERCOS (Serial Realtime Communication System) dienen.

Das SERCOS-Antriebsprofil definiert Kommunikationsobjekte, die in Antriebsgeräten vorhanden sein können. Diese Kommunikationsobjekte werden Parameter genannt. Die vorhandenen Parameter lassen sich in Funktionsgruppen (FGs) einteilen, wie sie in Anhang A aufgelistet sind.

Die in 7.5 beschriebenen vier Basisklassen von Antriebsgeräten implementieren unterschiedliche Funktionsgruppen innerhalb des Antriebsprofiles aus diesen Funktionsgruppen.

Ein Gerät, das die Klasse „Antrieb mit Momentenschnittstelle“ implementiert, muss auch folgende Funktionsgruppen und die darin enthaltenen Kommunikationsobjekte implementieren:

- FG Amplifier
- FG Motor
- FG Scaling
- FG Drive State Machine
- FG Operation Modes
- FG Torque Mode
- FG Torque Loop
- FG Torque Limit

Ein Gerät, das die Klasse „Antrieb mit Geschwindigkeitsschnittstelle“ implementiert, muss auch folgende Funktionsgruppen und die darin enthaltenen Kommunikationsobjekte implementieren:

- FG Amplifier
- FG Motor

- FG Scaling
- FG Drive State Machine
- FG Operation Modes
- FG Velocity Mode
- FG Velocity Loop
- FG Velocity Limit

Ein Gerät, das die Klasse „Antrieb mit Lageschnittstelle“ implementiert, muss auch folgende Funktionsgruppen und die darin enthaltenen Kommunikationsobjekte implementieren:

- FG Amplifier
- FG Motor
- FG Scaling
- FG Drive State Machine
- FG Operation Modes
- FG Position Mode
- FG Position Loop
- FG Position Limit
- FG Control Unit Homing
- FG Motor Feedback

Ein Gerät, das die Klasse „Antrieb mit Positionierschnittstelle“ implementiert, muss auch folgende Funktionsgruppen und die darin enthaltenen Kommunikationsobjekte implementieren:

- FG Amplifier
- FG Motor
- FG Scaling
- FG Drive State Machine
- FG Operation Modes

- FG Position Mode
- FG Position Loop
- FG Position Limit
- FG Velocity Limit
- FG Acceleration Limit
- FG Control Unit Homing
- FG Motor Feedback
- FG Drive Positioning

Es lässt sich auf diese Weise ein Klassendiagramm aufstellen, welches die einzelnen hier genannten Klassen innerhalb der Antriebsgeräte darstellt. Dieses Klassendiagramm ist in Bild 9.2 gezeigt.

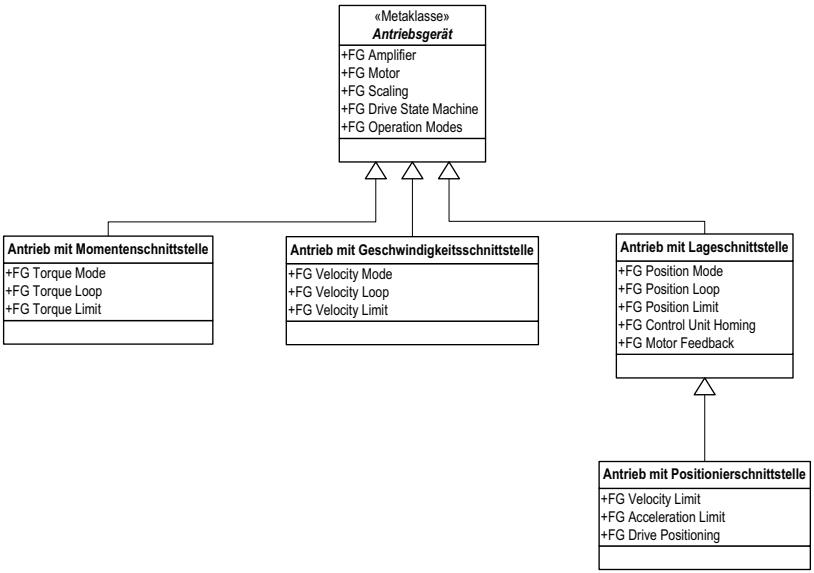


Bild 9.2: Klassendiagramm Antriebsgeräte

9.3 SDDML und SPDML

Auf Basis der in Kapitel 8 abgeleiteten Modelle entstanden eine Gerätebeschreibungssprache SDDML (SERCOS Device Description Markup Language) /97/ und eine Profilbeschreibungssprache SPDML (SERCOS Profile Description Markup Language) /98/ für den SERCOS III.

Diese werden von Herstellern von SERCOS III-Geräten benutzt, um ihre Geräte und deren Funktionen entsprechend der in SERCOS vorhandenen Profile zu beschreiben. Die beiden Sprachen sind Teil der offiziellen SERCOS III-Spezifikation.

9.4 Umsetzung in Engineeringwerkzeugen

Umsetzung fanden die entwickelten Methoden und Sprachen zur Beschreibung der Geräte und Profile in den Engineeringwerkzeugen der Firmen 3S und der Bosch Rexroth AG.

Bild 9.3 zeigt das Engineeringwerkzeug IndraWorks der Firma Bosch Rexroth AG bei der Interpretation einer Gerätebeschreibungsdatei für ein SERCOS III E/A-Gerät, dessen Beschreibung die Gerätebeschreibungssprache SDDML verwendet.

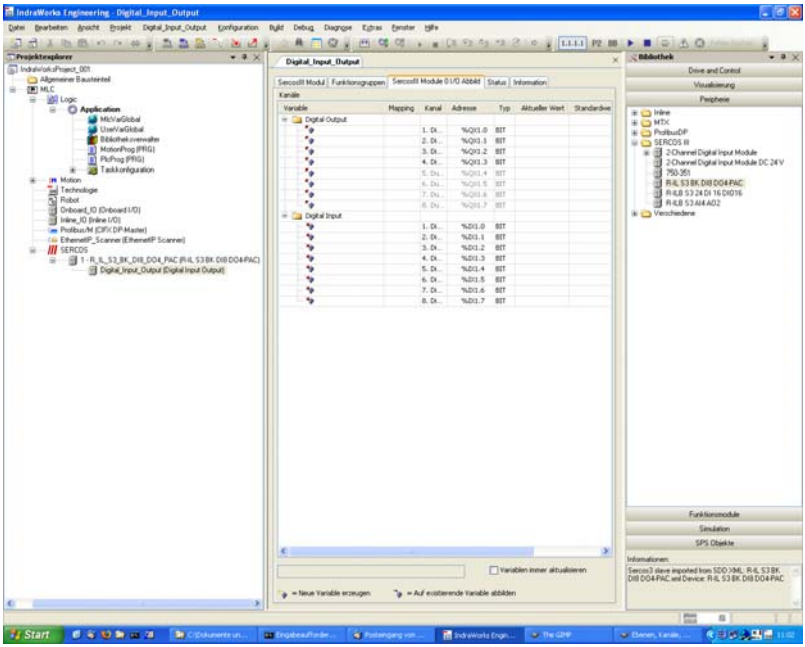


Bild 9.3: Oberfläche des Engineeringwerkzeugs IndraWorks

9.5 Zusammenfassung

Mittels der in diesem Kapitel gezeigten Umsetzungen wird ein Nachweis der in der vorliegenden Arbeit beschriebenen Methoden und Ergebnisse in der Praxis erbracht. Es erfolgte eine Umsetzung in kommerziellen Engineeringwerkzeugen von zwei Firmen. Des Weiteren wurden mittels der abgeleiteten Methode ein Geräte- und ein Profilbeschreibungsformat für Automatisierungsgeräte und deren Funktionen entwickelt, die in der industriellen Praxis Einsatz finden. Außerdem fand die Methode Anwendung bei der Transformierung von Gerätebeschreibungssprachen untereinander.

10 Zusammenfassung und Ausblick

10.1 Zusammenfassung

Das immer wieder geforderte „Plug and Produce“ ist heutzutage in der Automatisierungstechnik immer noch nicht erfüllt. Dies liegt unter anderem daran, dass die Kommunikationsmedien, die in der Automatisierungstechnik verwendet werden, komplexe und vielschichtige Schnittstellen zwischen den beteiligten Geräten bereitstellen. Die Schnittstelle selber hängt dabei nicht nur von dem verwendeten Kommunikationsmedium, sondern auch von der verwendeten Steuerungsarchitektur ab und wie zentral oder dezentral diese angelegt ist.

Um dem Ziel eines „Plug and Produce“ einen Schritt näher zu kommen, wird ein durchgängiges Informationsmodell für Automatisierungsgeräte benötigt, welches eine funktionale Sicht auf beteiligte Geräte und die durch sie bereitgestellten Kommunikationsobjekte erlaubt.

Ein Vergleich der in der Automatisierungstechnik existierenden Gerätebeschreibungssprachen und der zugrunde liegenden Informationsmodelle zeigte, dass keines der betrachteten Konzepte in der Lage ist, funktionale Zusammenhänge und deren Abhängigkeiten darzustellen.

Daher wurde in der vorliegenden Arbeit ein Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten erarbeitet. Mit diesem Modell lassen sich funktionale Abhängigkeiten auch geräteübergreifend beschreiben. Auf diese Art wird es möglich, bereits offline in einem Engineeringwerkzeug eine funktionale Sicht bereitzustellen. Diese kann dazu dienen, auf einer funktionalen Ebene zu testen, ob die durch eine Applikation geforderten Kommunikationsobjekte durch ausgewählte Geräte bereit gestellt werden können.

Im Rahmen der Arbeit wurden zwei Methoden abgeleitet. Die eine Methode dient dazu, Beschreibungssprachen zu vergleichen, indem das zugrunde liegende Informationsmodell abgeleitet wird. Die Beschreibungssprachen werden dann auf Modellebene verglichen.

Aufbauend auf dem Vergleich von Beschreibungssprachen können Abbildungsmöglichkeiten zwischen Beschreibungssprachen erarbeitet werden. Es ist auf diese Weise möglich, die Relationen zwischen Objekten in verschiedenen Beschreibungssprachen herzustellen und auch herauszufinden, welche Objekte nicht abgebildet werden können.

Die andere Methode ermöglicht es, Kommunikationsobjekte zu funktionalen Gruppen zusammenzufassen. Des Weiteren können mit ihr mögliche spezifikationskonforme Implementierungsgrade in Geräten abgeleitet werden. Die zweite Methode erlaubt zudem, potentielle Fehler bei der Erstellung einer Spezifikation aufzufinden, indem sie hilft, Widersprüche bezüglich der Beziehungen von Kommunikationsobjekten untereinander zu finden. Das Vorgehen beruht dabei auf der Graphentheorie und kann mittels Matrizenrechnung automatisiert werden.

Die Methoden und Ergebnisse dienten als Grundlage für die Erstellung eines spezifischen Informationsmodells für Geräte eines Ethernet-basierten Feldbusses und flossen auch in die Arbeit der Arbeitsgruppen dieses Feldbusses ein. Zudem fanden die Methoden und Ergebnisse Umsetzung in kommerziellen Engineeringwerkzeugen.

10.2 Ausblick

In einem weiteren Schritt ist es denkbar, die in dieser Arbeit hergeleiteten Ergebnisse nicht nur für die Konfiguration und Parametrierung von Maschinen und Anlagen und deren Komponenten zu benutzen, die offline in einem Engineeringwerkzeug erfolgt. Es muss untersucht werden, ob die entstandenen Informationsmodelle dazu dienen können, auch eine Konfiguration oder Rekonfiguration einer Maschine oder Anlage im laufenden Betrieb, also online und automatisch, zu ermöglichen. Dies würde zu einem wirklichen „Plug and Produce“ für die Automatisierungstechnik führen.

Literaturverzeichnis

/1/ Dietz, P.: Baukastensystematik und methodisches Konstruieren im Werkzeugmaschinenbau, Werkstatt und Betrieb 116, München, Carl Hanser Verlag.

/2/ Wurst, K.-H.; Kircher, C.; Seyfarth, M.: Conception of Reconfigurable Machining Systems – Design of Components and Controllers In: Bramley, A. (Hrsg.): Advances in Integrated Design and Manufacturing in Mechanical Engineering. Dordrecht, Springer, 2005.

/3/ Korajda, I.; Seyfarth, M.; Pritschow, G.: Disziplinübergreifende Baukastensysteme. Konfigurieren von Fertigungseinrichtungen aus einem disziplinübergreifenden Baukasten. wt Werkstattstechnik online 94, 2004.

/4/ Kircher, C.; Seyfarth, M.; Wurst, K.-H.: Modellbasiertes Rekonfigurieren von Werkzeugmaschinen. wt Werkstattstechnik online 94 (2004).

/5/ Lüder, A.: Industrielle Verkabelung im Umbruch: vom Feldbus zum Industrial Ethernet. Industrial Ethernet: Ethernet-basierte Automatisierungsprotokolle. Würzburg, Vogel Industrie Medien, 2006.

/6/ Rathjen, O.: Neue Möglichkeiten der Anlagenarchitektur durch Profinet, Tagungsband Fertigungstechnisches Kolloquium Stuttgart (FTK 2006), Stuttgart, Gesellschaft für Fertigungstechnik, 2006.

/7/ Wenzel, P.: Vertikale Integration durch alle Ebenen, A&D Kompendium 2008/2009, München, publish-industry Verlag GmbH, 2008.

/8/ Wurst, K.-H.; Heisel, U.; Kircher, C. : (Re)konfigurierbare Werkzeugmaschinen - notwendige Grundlage für eine flexible Produktion. wt Werkstattstechnik online, Jahrgang 96 (2006).

/9/	Lechler, A.; Schmitz, S.; Kircher, C.; Wurst K.-H.:	Flexible Reconfiguration and Maintenance through Mechatronic Moduls in Manufacturing Systems, ICPR, 19th International Conference in Production Research, Valparaiso Chile, 2007.
/10/	Kircher, C.; Meitzner, M.; Heisel, W.; Wurst, K.-H.	Wandelbare, zielvariable Bearbeitungssystem. Industrie Management 20, Berlin, Gito mbH Verlag für Industrielle Informationstechnik und Organisation, 2004.
/11/	Kircher, C.; Wurst, K.-H.:	Modulkonzept für rekonfigurierbare Bearbeitungs- und Steuerungssysteme.
/12/	N.N.	Guidelines for Packaging Machinery Automation Version 3.1 OMAC, 2006.
/13/	Brinkschulte, U.:	Mikrorechner Technik 2, Vorlesungsmanuskript am Institut für Prozessrechentechnik, Automation und Robotik, Karlsruhe, Sommersemester 2007.
/14/	ISO 7548	Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model.
/15/	IEC 61158	Digital data communications for measurement and control - Fieldbus for use in industrial control systems.
/16/	Wiesmüller, R.:	Rechnernetze II, Vorlesungsmanuskript am Lehrstuhl Betriebs-systeme und verteilte Systeme, Siegen, Wintersemester 2008.
/17/	IEC 8802-3	Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements, Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications Berlin, Köln: Beuth Verlag, 2000.

/18/ IEC 61158-4 Digital data communications for measurement and control - Fieldbus for use in industrial control systems, Part 4: Data-link layer protocol specification, Berlin, Köln: Beuth Verlag, 2003.

/19/ IEC 61158-6 Digital data communications for measurement and control - Fieldbus for use in industrial control systems, Part 6: Application layer protocol specification, Berlin, Köln: Beuth Verlag, 2003.

/20/ IEC 61800-7-2 Adjustable speed electrical power drive systems - Part 7-2: Generic interface and use of profiles for power drive systems - Profile specifications, Berlin, Köln: Beuth Verlag, 2006.

/21/ Postel, J.: RFC 791: Internet Protocol, Marina del Rey, 1981.

/22/ Postel, J.: RFC 793: Transmission Control Protocol, Marina del Rey, 1981

/23/ Postel, J.: RFC 768: User Datagram Protocol, Marina del Rey, 1980.

/24/ Postel, J.; Reynolds, J.: RFC 959: File Transfer Protocol, Marina del Rey, 1985.

/25/ Fielding, R.; et. al.: RFC 2616: Hypertext Transfer Protocol - HTTP/1.1, Cambridge, 1999.

/26/ N.N. IAONA Handbook Industrial Ethernet. Magdeburg: Industrial Automation Open Networking Alliance e.V., 2005.

/27/ Klemm, P.: Grundlagen der Prozessrechentechnik und der Softwaretechnik Vorlesungsmanuskript am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen Stuttgart, Wintersemester 2007.

/28/ DIN 19226-5 Leittechnik; Regelungstechnik und Steuerungstechnik; Teil 5: Funktionelle Begriffe, Berlin, Köln: Beuth Verlag, 1994.

/29/ Haubelt, A.; Verl, A.: Der Weg zur automatischen Generierung von Simulationsmodellen aus mechatronischen Baukästen. In: Brecher, Chr. (Hrsg.): ABS-Treffen 2008. Fortschritt-Berichte VDI, Reihe 2, Fertigungstechnik.

/30/ Klemm, P; Rüdele, H.; Weimer, T.: Durchgängiges Engineering von mechatronischen Systemen. Tagungsband Fertigungstechnisches Kolloquium Stuttgart (FTK 2006), Stuttgart, Gesellschaft für Fertigungstechnik, 2006.

/31/ Henning, K.: Kommunikation und Organisationsentwicklung II, Vorlesungsmanuskript am Lehrstuhl Informatik im Maschinenbau, Aachen, Sommersemester 2007.

/32/ Wimmer, M.: Verwaltungsinformatik I, Vorlesungsmanuskript am Institut für Wirtschafts- und Verwaltungsinformatik, Koblenz, Sommersemester 2006.

/33/ DIN EN ISO 9004 Qualitätsmanagementsysteme - Leitfaden zur Leistungsverbesserung.

/34/ DIN ISO 15226 Technische Produktdokumentation - Lebenszyklusmodell und Zuordnung von Dokumenten.

/35/ DIN 6789-1 Dokumentationssystematik; Teil 1: Aufbau Technischer Produktdokumentationen, Berlin, Köln: Beuth Verlag, 1990.

/36/ Zacher, S.: Automatisierungstechnik kompakt, Wiesbaden: Vieweg Verlagsgesellschaft, 2000.

/37/ Staudt, S.: Spezifikation und Konformitätstest zur Interoperabilität von automatisierten Produktionsmaschinen, Dissertation ISW 162 Heimsheim: Joost Jetter Verlag, 2006.

/38/ IEC 61800-7-3 Adjustable speed electrical power drive systems - Part 7-3: Generic interface and use of profiles for power drive systems - Mapping of profiles to network technologies, Berlin, Köln: Beuth Verlag, 2006.

/39/ König, H.: Protocol Engineering - Prinzip, Beschreibung und Entwicklung von Kommunikationsprotokollen, Stuttgart, Leipzig, Wiesbaden: Teubner Verlag, 2003.

/40/ König, H.: Protocol Engineering, Vorlesungsmanuskript am Institut für Informatik, Cottbus, Wintersemester 2008.

/41/ Simon, R.: Gerätemodell zur Feldinstrumentierung von Verteilten Automatisierungssystemen (Device Model for Field Instrumentation in Distributed Control Systems), at - Automatisierungstechnik 50 (2002) München: Oldenbourg Wissenschaftsverlag GmbH, 2002.

/42/ Simon, R.: Methoden zur Feldinstrumentierung von Verteilten Automatisierungssystemen. Dissertation, Forschungsbericht VDI Reihe 8, Nr. 918, VDI Verlag Düsseldorf, 2001.

/43/ Riedewald, G; Małuszyński, J; Dembiński, P.: Formale Beschreibung von Programmiersprachen, München, Wien: Oldenburg Verlag, 1983.

/44/ Glinz, M.: Requirements Engineering I, Vorlesungsmanuskript am Institut für Informatik, Zürich, Wintersemester 2006/2007.

/45/ ISO 14977 Information technology. Syntactic Metalanguage. Extended BNF.

/46/ Dausend, S.: Man spricht Profinet, iee 49. Jahrgang (2004), Heidelberg: Hüthig GmbH, 2004.

/47/ Etschberger, K.: Controller Area Network. Grundlagen, Protokolle, Bausteine, Anwendungen. München: Carl Hanser Verlag, 2001.

/48/ Müller, M.: Goldene Brücken gebaut, iee 49. Jahrgang (2004), Heidelberg: Hüthig GmbH, 2004.

/49/ N.N. DRIVECOM Device Description Version 1.1, Bloomberg, DRIVECOM Nutzergruppe e.V., 2002.

/50/ ISO 15745-3 Industrial automation systems and integration – Open systems application integration framework -- Part 3: Reference description for IEC 61158-based control systems.

/51/ ISO 154745-4 Industrial automation systems and integration – Open systems application integration framework -- Part 4: Reference description for Ethernet-based control system.

/52/ Zirkler, A.: Automatische Erstellung gerätespezifischer Testfälle zur Zertifizierung von Gerätebeschreibungen, Dissertation TU München, Göttingen: Sierke Verlag, 2008.

/53/ Appelrath, H; Ludewig, J.: Skriptum Informatik 5. Auflage, Stuttgart, Leibzig: B.G. Teubner, 2000.

/54/ Henning, K.: Informatik im Maschinenbau. Berlin: Springer Verlag, 1996.

/55/ Henkel, J.: Software-Engineering for Embedded Systems, Vorlesungsmanuskript am Department of Computer Science Karlsruhe, Wintersemester 2008/2009.

/56/ Jodlbauer, H.: Produktionsoptimierung: Wertschaffende sowie kundenorientierte Planung und Steuerung, 2. Auflage Berlin: Stringer, 2008.

/57/ Litto, M.; Korajda, I.; Mangold, C.: Baukastenbasiertes Engineering mit Föderal. Ein Leitfaden für Maschinen und Anlagenbauer. Frankfurt: VDMA, 2004.

/58/ N.N. Artikel Initialisierungsdatei. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 6. November 2007, 17:53 UTC. URL: http://de.wikipedia.org/w/index.php?title=Initialisierungsdatei&oldid=38680389 (Abgerufen: 4. Dezember 2007, 12:56 UTC).

/59/ Ray, T.: Einführung in XML, Köln: O'Reilly Verlag, 2004.

/60/ Van der Vlist, E.: XML Schema, Köln: O'Reilly Verlag, 2003.

/61/ ISO 19757-2 Information technology - Document Schema Definition Language – Part 2 Regular-grammar-based validation.

/62/ ISO 19757-3 Information technology - Document Schema Definition Language – Part 2 Regular-grammar-based validation.

/63/ Früh, M.; Häuser, P.; Marks, M.: Bewertung von XSLT – Prozessoren. Fachstudie 32 am Institut für Parallele und Verteilte Systeme (IPVS) Abteilung Anwendersoftware (AS) an der Universität Stuttgart, Stuttgart, 2004.

/64/ Pritschow, G.: Einführung in die Steuerungstechik, München, Wien: Carl Hanser Verlag, 2006.

/65/ Braune, A.; Wollschläger, M.: Serie – XML in der Automation, Computer & Automation Ausgaben 08-11/2007.

/66/ N.N. Specification for PROFIBUS Device Description and Device Integration, Volume 1: GSD Version 5.04, Karlsruhe: Profibus Nutzerorganisation e.V., 2005.

/67/ N.N. GSDML Specification for PROFINET IO Version 2.10, Karlsruhe: Profibus Nutzerorganisation e.V., 2006.

/68/ Iwanitz, F.: Alles bleibt anders - Migration von DP zu Profinet-IO IEE Ausgabe 11-2004.

/69/ Gedenk, T.: Busunabhängige Gerätebeschreibungen mit XML. SPS-Magazin; 2005.

/70/ Augustin, M.; Polzer, K.; Ott, W.: Electronic Device Description Language – Basis für eine einheitliche und plattformunabhängige Gerätebedienung Erlangen: Siemens AG, 2000.

/71/ IEC 61804-3 Function blocks (FB) for process control - Part 3: Electronic Device Description Language (EDDL), Berlin, Köln: Beuth Verlag, 2006.

/72/ N.N. FDT Interface Specification V1.2.1, Diegem: FST Joint Interest Group, 2004.

/73/ N.N. Device Type Manager (DTM) Style Guide V1.0, Diegem: FST Joint Interest Group, 2005.

/74/ N.N. Mit PACTware effizient konfigurieren. Pfinztal: PACTware Consortium e. V., 2007.

/75/ N.N. Konfigurieren Parametrieren Diagnostizieren, Schwerin, Schillach, Duisburg: Trebing+Himstedt, Vega, Krohne, 2008.

/76/ N.N. FieldCare - Anlagen-Asset-Management, Weil am Rhein: Endress+Hauser, 2004.

/77/ N.N. Datenblatt fdtCONTAINER
St. Georgen: M&M, 2005.

/78/ N.N. Datenblatt Geräte-Management-Tool DSV4xx (SMART VISION), für intelligente Feldgeräte Zürich: ABB, 2002.

/79/ N.N. Datenblatt cifX – reduziert auf das Maximum,
Hattersheim: hilscher, 2007.

/80/ Amt für amtliche Veröffentlichungen der Europäischen Gemeinschaften Handbuch für die Bewertung der Sicherheit von Systemen der Informationstechnik (ITSEM),
Brüssel Luxemburg, 1994, ISBN 92-826-7078-2.

/81/ Nissanke, N.: Formal specification: techniques and applications,
London: Springer Verlag, 1999.

/82/ Drechsler, R.: Methoden der Verifikation, Vorlesungsmanuskript im Fachbereich Informatik, Bremen, Wintersemester 2008/2009.

/83/ Diestel, R.: Graphentheorie, Heidelberg: Springer Verlag, 2006.

/84/ Ehrig, H.; Mahr, B.; Cornelius, F.; Große-Rhode, M.; Zeitz, P.: Mathematisch-strukturelle Grundlagen der Informatik,
Berlin, Heidelberg: Springer Verlag, 1999.

/85/ Wagner, T.: Aktuelle Entwicklungen bei XML-basierten Geräte- und Systembeschreibungen in der Automation, Umdruck im Hauptseminar Fabrik- und Gebäudeautomation, Dresden: 2006.

/86/ Lechler, A.; Selig, A.; Verl, A. Konzept für ein Testwerkzeug zur schnellen Entwicklung von konformen ethernetbasierten Feldbusgeräten, Verl A., Bender K., Schumacher W., SPS/IPC/Drives, 27.-29. November 2007, Nürnberg, Berlin: VDE, 2007, S. 323-329.

/87/ N.N. Produktkatalog Komponenten und Systeme – AUTOMATION Blomberg: Phoenix Contact, 2008.

/88/ N.N. Produktkatalog product LOCATOR, Minden: Wago, 2008.

/89/ N.N. Produktkatalog Feldbuskomponenten, Busklemmen, Verl: Beckhoff, 2008.

/90/ Riefenstahl, U.: Elektrische Antriebstechnik, Stuttgart, Leibzig: Teubner, 2000.

/91/ N.N. General Overview and Architecture V.1.1.2.1.3, Süssen: SERCOS International e.V., 2009.

/92/ N.N. Generic Device Profile V.1.1.2.1.1 Süssen: SERCOS International e.V., 2009.

/93/ N.N. Communication V.1.1.2.1.7 Süssen: SERCOS International e.V., 2009.

/94/ N.N. Communication Profile V.1.1.2.1.1 Süssen: SERCOS International e.V., 2009.

/95/ N.N. Function specific Profile Drives V.1.1.2.2.13 Süssen: SERCOS International e.V., 2009.

/96/ N.N. Function specific Profile IO V.1.1.2.1.4 Süssen: SERCOS International e.V., 2009.

/97/ N.N. SERCOS device description V1.1.2.1.7.
Süssen: SERCOS International e.V., 2009.

/98/ N.N. SERCOS profile description V1.1.2.1.0
Süssen: SERCOS International e.V., 2009.

/99/ Selig, A.; Wetter, O.; Rüppel, W.: Innovatives Gerätemodell und zukunftsweisendes I/O-Geräte-profil für SERCOS III, Computer & Automation 11/2007.

/100/ Lutz, P.; Selig, A.: Sercos III - Steuerungsvernetzung mit Echtzeit-Ethernet, SPS-Magazin; Heft HMI Special; 2006

/101/ Selig, A.; Kircher, C.: Grundlegende Betrachtung zu Plug and Play-Konzepten im Maschinen- und Anlagenbau am Beispiel SERCOS III.
2. Rexroth Doktoranden Kolloquium, Lohr a. Main, 2006

Anhang A

Die Kommunikationsobjekte des SERCOS-Antriebsprofils lassen sich in folgende Funktionsgruppen (FGs) zusammenfassen:

- **FG Drive State Machine**: Diese Funktionsgruppe beinhaltet alle Parameter, die mit der Steuerung und der Anzeige des aktuellen Zustands der Zustandsmaschine des Antriebs verbunden sind.
- **FG Amplifier**: Diese Funktionsgruppe vereint die Parameter, die der Beschreibung des Antriebsverstärkers dienen.
- **FG Motor**: Alle Parameter, die der Identifikation des angeschlossenen Motors dienen, werden in dieser Funktionsgruppe zusammengefasst.
- **FG Brake**: Die Parameter, die die Motorbremse betreffen, befinden sich in dieser Funktionsgruppe.
- **FG Scaling**: In dieser Funktionsgruppe werden alle Parameter zusammengefasst, die die Skalierung der Antriebsdaten betreffen.
- **FG Mechanics**: In dieser Funktionsgruppe werden alle Parameter zusammengefasst, die die Mechanik der Achse betreffen.
- **FG Motor Feedback**: Alle Parameter, die den Motorgeber betreffen, werden in dieser Funktionsgruppe zusammengefasst.
- **FG External Feedback**: Die Parameter, die mit einem externen Geber verbunden sind, finden sich in dieser Funtionsgruppe.
- **FG Drive Warning**: In dieser Funktionsgruppe werden die Parameter zusammengefasst, die zur Steuerung von Warnungen bereitgestellt werden.
- **FG Torque Limit**: Alle Parameter, die zur Einstellung und Überwachung von Momentengrenzen genutzt werden, finden sich in dieser Funktionsgruppe.
- **FG Velocity Limit**: Die Parameter in dieser Funktionsgruppe werden dazu genutzt, um Geschwindigkeitsgrenzen zu definieren und zu überwachen.
- **FG Position Limit**: In dieser Funktionsgruppe werden die Parameter zusammengefasst, die zur Einstellung und Überwachung von Positionsgrenzen verwendet werden.

- **FG Acceleration Limit**: Alle Parameter, die zur Einstellung und Überwachung von Beschleunigungsgrenzen genutzt werden, finden sich in dieser Funktionsgruppe.
- **FG Temperature Limit**: In dieser Funktionsgruppe werden die Parameter zusammengefasst, die zur Einstellung und Überwachung von Temperaturgrenzen verwendet werden.
- **FG Voltage Limit**: Die Parameter in dieser Funktionsgruppe werden dazu genutzt, um Spannungsgrenzen zu definieren und zu überwachen.
- **FG Power Limit**: Alle Parameter, die zur Einstellung und Überwachung von Stromgrenzen genutzt werden, finden sich in dieser Funktionsgruppe.
- **FG Operation Modes**: Über die in dieser Funktionsgruppe zusammengefassten Parameter kann der Betriebsmodus des Antriebs und damit die durch den Antrieb bediente Schnittstelle beeinflusst werden.
- **FG Torque Mode**: Diese Funktionsgruppe fasst alle Parameter zusammen, die die Momentensollwertschnittstelle des Antriebs betreffen.
- **FG Torque Loop**: In dieser Funktionsgruppe werden die Parameter zusammengefasst, die zur Steuerung des Momentenregelkreises im Antrieb dienen.
- **FG Velocity Mode**: Die Parameter, die die Geschwindigkeitssollwert-schnittstelle betreffen, werden in dieser Funktionsgruppe zusammengefasst.
- **FG Velocity Loop**: Alle Parameter, die zur Steuerung des Geschwindigkeits-regelkreises dienen, finden sich in dieser Funktionsgruppe.
- **FG Position Mode**: Diese Funktionsgruppe fasst alle Parameter zusammen, die die Lagesollwertschnittstelle des Antriebs betreffen.
- **FG Position Loop**: In dieser Funktionsgruppe werden die Parameter zusammengefasst, die zur Steuerung des Lageregelkreises im Antrieb dienen.
- **FG Drive Interpolation**: Alle Parameter, die zur Steuerung des antriebsinternen Interpolators dienen, werden in dieser Funktionsgruppe zusammengefasst.
- **FG Drive Positioning**: In dieser Funktionsgruppe werden die Parameter, die die Positionierschnittstelle des Antriebs betreffen, zusammengefasst.
- **FG Drive Homing**: Diese Funktionsgruppe fasst die Parameter zusammen, die mit dem antriebsgeführten Referenzieren verbunden sind.

- **FG Control Unit Homing**: Alle Parameter, die das steuerungsgeführte Referenzieren betreffen, sind in dieser Funktionsgruppe zusammengefasst.
- **FG Modify Coordinate System**: Diese Funktionsgruppe vereint alle Parameter, die zur Veränderung des antriebsinteren Koordinatensystems verwendet werden.
- **FG Set Absolut Position**: Alle Parameter, die mit dem Setzen der absoluten Position verbunden sind, finden sich in dieser Funktionsgruppe.
- **FG Probe**: Diese Funktionsgruppe beinhaltet alle Parameter, die mit dem Betrieb von Messtastern verbunden sind.
- **FG Position Spindle**: Alle Parameter, die mit dem Betrieb des Antriebs als Spindel verbunden sind, finden sich in dieser Funktionsgruppe.
- **FG Drive Synchronous Operation**: Diese Funktionsgruppe vereinigt alle Parameter, die zu einem synchronen Betrieb mehrerer Antriebe notwendig sind.
- **FG Parameter Sets**: Alle Parameter die zum Ändern des Parametersatzes notwendig sind, werden in dieser Funktionsgruppe zusammengefasst.
- **FG Gear Engaging**: In dieser Funktionsgruppe werden alle Parameter zusammengefasst, die bei Wechsel der Getriebestufe benutzt werden.
- **FG Positive Stop**: Diese Funktionsgruppe vereinigt alle Parameter, die zum Fahren auf Festanschlag bei einem Antrieb gebraucht werden.
- **FG Park Axis**: Alle Parameter, die zum Stillsetzen eines Antriebs verwendet werden, befinden sich in dieser Funktionsgruppe.
- **FG Position Switch**: Die Parameter, die mit dem Betrieb von Software-Endschaltern verbunden sind, finden sich in dieser Funktionsgruppe.
- **FG Drive Diagnosis**: Alle Parameter, die antriebsspezifische Diagnose erlauben, befinden sich in dieser Funktionsgruppe.

Lebenslauf

Persönliches	Name	Andreas Selig
	Geburtstag	02.04.1977
	Geburtsort	Rinteln an der Weser
	Eltern	Heinz Selig Adelheid Selig, geb. Hanke
	Ehefrau	Corina Selig, geb. Briel
	Kinder	Jannek und Leni
Schulbildung	1987 bis 1989	Gymnasium Andreanum Hildesheim
	1989 bis 1996	Goethe-Gymnasium Ludwigsburg
Ersatzdienst	1996 bis 1997	Johanniter-Unfallhilfe e.V. Ludwigsburg
Studium	1997 bis 2003	Studium der Elektro- und Informationstechnik an der Universität Stuttgart
	2001 bis 2002	Studium der Informatik an der École Supérieur d'Électricité in Rennes, Frankreich
Berufstätigkeit	2003 bis 2008	Wissenschaftlicher Mitarbeiter am Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart (ISW)
	Seit 2008	Entwicklungsingenieur bei Bosch Rexroth Electric Drives and Controls GmbH im Bereich Antriebsentwicklung
	Seit 2010	Fachreferent Ethernet Strategie bei der Bosch Rexroth AG

ISW/IPA Forschung und Praxis

Berichte aus dem Institut für Steuerungstechnik der Werkzeugmaschinen und Fertigungseinrichtungen der Universität Stuttgart und dem Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA

Herausgegeben bis Band 57 von Prof. Dr.-Ing. G. Stute †
ab Band 58 von Prof. Dr.-Ing. Dr. h. c.mult. Dr.-Ing. E.h G. Pritschow
ab Band 161 von Prof. Dr.-Ing. A. Verl

Erschienen bei Springer-Verlag:

1	Schmid, D.:	Numerische Bahnsteuerung, 1973.
2	Schwegler, H.:	Fräsbearbeitung gekrümmter Flächen, 1972.
3	Eisinger, J.:	Numerisch gesteuerte Mehrachsenfräsmaschinen, 1972.
4	Nann, R.:	Rechnersteuerung von Fertigungseinrichtungen, 1972.
5	Augsten, G.:	Zweiachsige Nachformeinrichtungen, 1972.
6	Karl, B.:	Die Automatisierung der Fertigungsvorbereitung durch NC-Programmierung, 1972.
7	Eitel, H.:	NC-Programmiersystem, 1973.
8	Knorr, E.:	Numerische Bahnsteuerung zur Erzeugung von Raumkurven auf rotationssymetrischen Körpern, 1973.
9	Bumiller, S.:	Viskohydraulischer Vorschubantrieb, 1974.
10	Maier, K.:	Grenzregelung an Werkzeugmaschinen, 1974.
11	Waelkens, J.:	NC-Programmierung, 1974.
12	Bauer, E.:	Rechnerdirektsteuerung von Fertigungseinrichtungen, 1975.
13	König, H.:	Entwurf und Strukturtheorie von Steuerungen für Fertigungseinrichtungen, 1976.
14	Damsohn, H.:	Fünfachsiges NC-Fräsen, 1976.
15	Jetter, H.:	Programmierbare Steuerungen, 1976.
16	Henning, H.:	Fünfachsiges NC-Fräsen gekrümmter Flächen, 1976.
17	Boelke, K.:	Analyse und Beurteilung von Lagesteuerungen für numerisch gesteuerte Werkzeugmaschinen, 1977.

18 Götz, F.-R.: Regelsystem mit Modellrückkopplung für variable Streckenverstärkung, 1977.

19 Tränkle, H.: Auswirkungen der Fehler in den Positionen der Maschinenachsen beim fünfachsigen Fräsen, 1977.

20 Stof, P.: Untersuchungen über die Reduzierung dynamischer Bahnabweichungen bei numerisch gesteuerten Werkzeugmaschinen, 1978.

21 Wilhelm, R.: Planung und Auslegung des Materialflusses flexibler Fertigungssysteme, 1978.

22 Kappen, N.: Entwicklung und Einsatz einer direkten digitalen Grenzregelung für eine Fräsmaschine mit CNC, 1979.

23 Klug, H. G.: Integration automatisierter technischer Betriebsbereiche, 1978.

24 Binder, D.: Interpolation in numerischen Bahnsteuerungen, 1979.

25 Klingler, O.: Steuerung spanender Werkzeugmaschinen mit Hilfe von Grenzregeleinrichtungen, 1979.

26 Schenke, L.: Auslegung einer technologisch-geometrischen Grenzregelung für die Fräsbearbeitung, 1979.

27 Wörn, H.: Numerische Steuersysteme – Aufbau und Schnittstellen eines Mehrprozessorsteuersystems, 1979.

28 Osofisan, P. B.: Verbesserung des Datenflusses beim fünfachsigen NC-Fräsen, 1979.

29 Berner, J.: Verknüpfung fertigungstechnischer NC-Programmiersysteme, 1979.

30 Böbel, K.-H.: Rechnerunterstützte Auslegung von Vorschubantrieben, 1979.

31 Dreher, W.: NC-gerechte Beschreibung von Werkstücken in fertigungstechnisch orientierten Programmiersystemen, 1980.

32 Schurr, R.: Rechnerunterstützte Projektsteuerung hydrostatischer Anlagen, 1981.

33 Sielaff, W.: Fünfachsiges NC-Umfangfräsen verwundener Regelflächen. Beitrag zur Technologie und Teileprogrammierung, 1981.

34 Hesselbach, J.: Digitale Lageregelung an numerisch gesteuerten Fertigungseinrichtungen, 1981.

35 Fischer, P.: Rechnerunterstützte Erstellung von Schaltplänen am Beispiel der automatisierten Hydraulikplanzeichnung, 1981.

36 Ackermann, U.: Rechnerunterstützte Auswahl elektrischer Antriebe für spanende Werkzeugmaschinen, 1981.

37 Döttling, W.: Flexible Fertigungssysteme – Steuerung und Überwachung des Fertigungsablaufs, 1981.

38 Firnau, J.: Flexible Fertigungssysteme – Entwicklung und Erprobung eines zentralen Steuersystems, 1982.

39 Herrscher, A.: Flexible Fertigungssysteme – Entwurf und Realisierung prozeßnaher Steuerungsfunktionen, 1982.

40 Spieth, U.: Numerische Steuersysteme – Hardwareaufbau und Ablaufsteuerung eines Mehrprozessorsteuersystems, 1982.

41 Schimmele, A.: Rechnerunterstützter Entwurf von Funktionssteuerungen für Fertigungseinrichtungen, 1982.

42 Sanzenbacher, M.: NC-gerechte Beschreibung von Werkstücken mit gekrümmten Flächen, 1982.

43 Walter, W.: Interaktive NC-Programmierung von Werkstücken mit gekrümmten Flächen, 1982.

44 Huan, J.: Bahnregelung zur Bahnerzeugung an numerisch gesteuerten Werkzeugmaschinen, 1982.

45 Erne, H.: Taktile Sensorführung für Handhabungseinrichtungen – Systematik und Auslegung der Steuerungen, 1982.

46 Plasch, D.: Numerische Steuersysteme – Standardisierte Softwareschnittstellen in Mehrprozessor-Steuersystemen, 1983.

47 Wang, Z. L.: NC-Programmierung – Maschinennaher Einsatz von fertigungstechnisch orientierten Programmiersystemen, 1983.

48 Schwager, J.: Diagnose steuerungsexterner Fehler an Fertigungseinrichtungen, 1983.

49 Klemm, P.: Strukturierung von flexiblen Bediensystemen für numerische Steuerungen, 1984.

50 Runge, W.: Simulation des dynamischen Verhaltens elektrohydraulischer Schaltungen – Einsatz von geräteorientierten, universellen Simulationsbausteinen, 1984.

51 Steinhilber, H.: Planung und Realisierung von Werkzeugversorgungssystemen für die NC-Bearbeitung, 1984.

52 Ohnheiser, R.: Integrierte Erstellung numerischer Steuerdaten für flexible Fertigungssysteme, 1984.

53 Keppeler, M.: Führungsgrößenerzeugung für numerisch bahngesteuerte Industrieroboter, 1984.

54 Kohler, P.: Automatisiertes Messen mit NC-Werkzeugmaschinen, 1985.

55 Rieger, K.-H.: Rechnerunterstützte Projektierung der Hardware und Software von Speicherprogrammierten Steuerungen, 1985.

56 Vogt, G.: Digitale Regelung von Asynchronmotoren für numerisch gesteuerte Fertigungseinrichtungen, 1985.

57 Chmielnicki, S.: Flexible Fertigungssysteme – Simulation der Prozesse als Hilfsmittel zur Planung und zum Test von Steuerprogrammen, 1985.

58 Renn, W.: Struktur und Aufbau prozeßnaher Steuergeräte zur Verkettung in flexiblen Fertigungssystemen, 1986.

59 Harig, K.: Quantisierung im Lageregelkreis numerisch gesteuerter Fertigungseinrichtungen, 1986.

60 Frank, H.: Programmier- und Überwachungsfunktionen für teileartbezogene NC-Werkzeugmaschinen, 1986.

61 Möller, H.: Integrierte Überwachungs- und Diagnose-Systeme für numerische Steuerungen, 1986.

62 Fink, H.: Einsatz speicherprogrammierbarer Steuerungen in der Fertigungstechnik, 1986.

63 Fleckenstein, J.: Zustandsgraphen für SPS – Grafikunterstützte Programmierung und steuerungsunabhängige Darstellung, 1987.

64 Wagner, E.: Steuerungen von Koordinatenmeßgeräten mit schaltenden und messenden Tastsystemen, 1987.

65 Grimm, W.: Diagnosesystem für steuerungsperiphere Fehler an Fertigungseinrichtungen, 1987.

66 Swoboda, W.: Digitale Lageregelung für Maschinen mit schwach gedämpften schwingungsfähigen Bewegungsachsen, 1987.

67 Gruhler, G.: Sensorgeführte Programmierung bahngesteuerter Industrieroboter, 1987.

68 Walker, B.: Konfigurierbarer Funktionsblock Geometriedatenverarbeitung für numerische Steuerungen, 1987.

69 Mayer, J.: Werkzeugorganisation für flexible Fertigungszellen und -systeme, 1988.

70 Lederer, R.: Programmierung von NC-Drehmaschinen mit mehreren Werkzeugschlitten, 1988.

71 Häberle, G.: NC-Musterprogrammierung für rechnerintegrierte Textilfertigung, 1988.

72 Pfeiffer, D.: Kompensation thermisch bedingter Bearbeitungsfehler durch prozeßnahe Qualitätsregelung, 1988.

73 Schmidt, W.: Grafikunterstütztes Simulationssystem für komplexe Bearbeitungsvorgänge in numerischen Steuerungen, 1988.

74 Egner, M.: Hochdynamische Lageregelung mit elektrohydraulischen Antrieben, 1988.

75 Schittenhelm, W.: Konfigurierbares Bedienungssystem für Steuerungen an Fertigungseinrichtungen, 1988.

76 Scheifele, D.: Grafisch dynamische Simulation des Bearbeitungsvorgangs für Doppelschlittendrehmaschinen, 1988.

77 Keuper, G.: Automatisierte Identifikation der Streckenparameter servohydraulischer Vorschubantriebe, 1989.

78 Kayser, K.-H.: Kollisionserkennung in numerischen Steuerungen mit der Distanzfeldmethode, 1989.

79 Viefhaus, R.: Fräsergeometriekorrektur in Numerischen Steuerungen für das fünfachsige Fräsen, 1989.

80 Zirbs, J.: Fertigungsgerechte Aufbereitung von Flächenverbänden bei der NC-Programmierung im Formenbau, 1989.

81 Ruoff, W.: Optische Sensorsysteme zur On-line-Führung von Industrierobotern, 1989.

82 Jantzer, M.: Bahnverhalten und Regelung fahrerloser Transportsysteme ohne Spurbindung, 1990.

83 Schumacher, H.: Einheitliche Programmierung von Automatisierungskomponenten roboterbestückter Bearbeitungs- und Montagezellen, 1991.

84 Schimonyi, J.: NC-Programmierung für das Werkzeugschleifen, 1991.

85 Wurst, K.-H.: Flexible Robotersysteme – Konzeption und Realisierung modularer Roboterkomponenten, 1991.

86 Hagl, R.: Erhöhung der Verfügbarkeit von Vorschubantrieben mit selbstanpassender Lageregelung, 1991.

87 Krebser, G.: Betriebssystem für NC mit einheitlichen Schnittstellen, 1992.

88 Lei, W.-T.: Flächenorientierte Steuerdatenaufbereitung für das fünfachsige Fräsen, 1992.

89 Diehl, G.: Steuerungsperipheres Diagnosesystem für Fertigungseinrichtungen auf Basis überwachungsgerechter Komponenten, 1992.

90 Nepustil, U.: Offene NC-Schnittstellen zur Korrektur von Fertigungsfehlern, 1992.

91 Bauder, M.: Konfigurierbare Robotersteuerung mit allgemeiner Transformation, 1992.

92 Philipp, W.: Regelung mechanisch steifer Direktantriebe für Werkzeugmaschinen, 1992.

93 Härdtner, G. M.: Wissensstrukturierung in Diagnoseexpertensystemen für Fertigungseinrichtungen, 1992.

94 Wiedmann, H.: Objektorientierte Wissensrepräsentation für die modellbasierte Diagnose an Fertigungseinrichtungen, 1993.

95 Rudloff, H.: Hochgenaue Konturerzeugung bei Bewegungsachsen mit einer dominanten mechanischen Resonanzstelle, 1993.

96 Brantner, K.: Adaptierbares Leitsteuerungssystem für flexible Produktionssysteme, 1993.

97 Kugler, W.: Kommunikationsmechanismen für offene Numerische Steuerungssysteme, 1994.

98 Schnurr, B.: Elektrodynamisches Antriebssystem zur Unrundbearbeitung, 1994.

99 Schneider, J.: Fehlerreaktion mit Speicherprogrammierbaren Steuerungen – ein Beitrag zur Fehlertoleranz, 1994.

100 Siewert, U.: Systematische Erstellung adaptierbarer Leitsteuerungssoftware am Beispiel der Durchsetzungsplanung, 1994.

101 Heger, G. F. J.: Maschinenferner Qualitätsregelkreis in flexiblen Fertigungssystemen, 1994.

102 Hofmeister, W.: Objektorientiert strukturiertes Programmiersystem für NC-Mehrschlittenmaschinen, 1994.

103 Horn, A.: Optische Sensorik zur Bahnführung von Industrierobotern mit hohen Bahngeschwindigkeiten, 1994.

104 Rentschler, U.: Fehlertolerantes Präzisionsfügen, 1995.

105 Junghans, G.: Modulares grafikunterstütztes Simulationssystem für Bearbeitungs- und Handhabungsvorgänge, 1995.

106 Heller, J.: Sensorgestützte Bewegungserzeugung leitlinienloser Transprtfahrzeuge, 1995.

107 Wieland, E.: Anwendungsorientierte Programmierung für die robotergestützte Montage, 1995.

108 Ketterer, G.: Automatisierte Inbetriebnahme elektromechanischer, elastisch gekoppelter Bewegungsachsen, 1995.

109 Reibetanz, Th.: Situationsorientierte Bearbeitungsmodellierung zur NC-Programmierung, 1995.

110 Frager, O.: Durchgängige Programmierung von Fertigungszellen, 1996.

111 Ordenewitz, R.: Betriebsweite Bereitstellung von Werkzeuginformationen, 1996.

112 Daniel, C.: Dynamisches Konfigurieren von Steuerungssoftware für offene Systeme, 1996.

113 Angerbauer, R.: Anwenderorientierte Programmierung fahrerloser Transportsysteme, 1996.

114 Krauß, F.: Splineverarbeitung in numerischen Steuerungen für das fünfachsige Fräsen, 1996.

115 Schittenhelm, K.-M.: Einsatz vorgefilterter Führungsgrößen für Bewegungsachsen zur Bahnerzeugung, 1997.

116 Häberle, U.: Einheitliche Anwenderschnittstelle für Feldbussysteme, 1997.

117 Strassacker, D.: Testumgebung für die Implementierung und Inbetriebnahme eines adaptierbaren Leitsteuerungssystems, 1997.

118 Renz, B.: Hochdynamische Strahllagekorrektursysteme zur Erhöhung der Bahngenauigkeit von CO_2-Laserbearbeitungsmaschinen, 1997.

119 Itterheim, C.: Objektorientiertes Bearbeitungsmodell für Freiformflächen – Erstellung und maschinengebundene Modifikation –, 1997.

120 Müller, J.: Objektorientierte Softwareentwicklung für offene numerische Steuerungen, 1997.

121 Glöckler, M.: Verbesserung des Störverhaltens elektrohydraulischer lagegeregelter Zylinderantriebe, 1998.

122 Uhl, J.: Entwurfssystematik für ein dezentral strukturiertes, objektorientiertes Fertigungsleitsystem, 1998.

123 Hammann, G.: Modellierung des Abtragsverhaltens elastischer, robotergeführter Schleifwerkzeuge, 1998.

124 Scholich-Tessmann, W.: Direktantriebe für Industrieroboter, 1998.

125 Wagner, R.: Robotersysteme zum kraftgeführten Entgraten grobtolerierter Leichtmetallwerkstücke mit Fräswerkzeugen, 1998.

126 Anders, C.: Adaptierbares Diagnosesystem bei Transferstraßen, 1998.

127 Fahrbach, C.: Regelung hochdynamischer elektrischer Servo-Direktantriebe in Fertigungseinrichtungen, 1999.

128 Sperling, W.: Modulare Systemplattformen für offene Steuerungssysteme, 1999.

Erschienen bei Jost-Jetter Verlag:

129 Kehl, G.: Gestaltung von Formgedächtnis-Aktorsystemen für sensorgeführte Inspektionsgeräte, 1999.

130 Brandl, Th.: Anlageninformationssystem - Informationsmodell und Erstellungssystematik, 1999.

131 Gronbach, H.: Simulationswerkzeug für die Gestaltung modularer CO_2-Laserbearbeitungsmaschinen, 1999.

132 Lutz, R.: Softwaretechnik für Maschinennahe Steuerungsfunktionen bei Fertigungseinrichtungen, 1999.

133 Tran, T. L.: Allgemeine Transformation für Maschinen mit Parallelkinematiken, 2000.

134 Bretschneider, J.: Reglerselbsteinstellung für digital geregelte, elektromechanische Antriebssysteme an Werkzeugmaschinen, 2000.

135 Schoenenberg, M.: Zuverlässiger Fertigungsprozess bei Transferstraßen durch präventive Maßnahmen, 2000.

136 Uhl, A.: Flexibles Telerobotersteuerungssystem auf der Basis offener numerischer Steuerungen, 2000.

137 Rui Li: Agentenbasierte NC-Planung für die Komplettbearbeitung auf Dreh-/Fräszentren, 2001.

138 Wildermuth, D.: Bahnvorbereitung in numerischen Steuerungen für Parallelkinematiken, 2001.

139 Handel, D.: Werkergerechte NC-Programmierung zur Komplett-Schleifbearbeitung von Bohrwerkzeugen, 2001.

140 Kosiedowski, U.: Adaptive Vorsteuerverfahren für elektromechanische Bewegungsachsen an Werkzeugmaschinen, 2001.

141 Haug, K.: Laser-Lichtschnittsensorik für die Automatisierung von Metall-Schutzgasschweißprozessen, 2002.

142 Hohenadel, J.: Einheitliches Steuerungssystem für NC und RC, 2002.

143 Wälde, K.: Sicherstellung der Softwarequalität von Anwendungsmodulen und Systemplattformen in offenen Steuerungssystemen, 2002.

144 Litto, M.: Störungsinformationssystem – Informationsmodell und Erstellungssystematik, 2002.

145 Schweiker, A.: Offene numerische Steuerungen für prozeßabhängige Bearbeitungen – vereinheitlichte Struktur, Funktionen und Schnittstellen – , 2003.

146 Rempp, B.: Regelungstechnische Untersuchung durchsatzgeregelter Produktionssysteme, 2003.

147 Eppler, C.: Kompensation fremderregter Schwingungen in Antriebssystemen mit Umlaufgetrieben, 2003.

148 Weiner, M.: Wiederverwendungsgerechte Entwicklungssystematik von Feinplanungssoftware für die flexible Fertigung, 2004.

149 McCormac, St.: Lageregelung hydraulischer Manipulatoren unter Einsatz eines Ferraris-Sensors, 2004.

150 Lehner, W.-D.: Regelung von Vorschubachsen unter Verwendung der Relativbeschleunigung, 2005.

151 Lewek, J.: Adaptierbares Informationssystem zur Erstellung baukastenbasierter Fertigungseinrichtungen, 2005.

152 Laible, U.: Aufbau numerischer Steuerungssysteme für sicherheitskritische Anwendungen, 2005.

153 Bürger, Th.: Durchgängige analytische Qualitätssicherung für numerische Steuerungssoftware, 2005.

154 Brinzer, B.: Produktionsregelung für die variantenreiche Serienfertigung, 2005.

155 Heusinger, S.: STEP-NC-basierter Korrekturkreis für die Schlichtbearbeitung von Freiformflächen, 2005.

156 Kirchberger, R.: Verbesserte Auswertung inkrementeller Messsysteme durch schnelle Signal-Vorverarbeitung, 2005.

157 Conrath, M.: Systematische Gestaltung von frequenzadaptierbaren Ultraschall-Werkzeugsystemen zum Einsatz in fertigungstechnischen Prozessen, 2005.

158 Wadehn, W.: Gestaltung von Antriebssystemen für formadaptive Strukturen, 2005.

159 Horber, H.: Fugendetektion bei Lichtbogenschweißprozessen mit robuster Signalverarbeitung für optische Sensoren, 2006.

160 Dreyer, J.: Situative Informationsbereitstellung an Fertigungseinrichtungen Informationsmodell und Erstellungssystematik, 2006.

161 Fritz, S.: Rekonstruktion von Prozesskräften aus Antriebssignalen von Werkzeugmaschinen, 2006.

162 Staudt, S.: Spezifikation und Konformitätstest zur Interoperabilität von automatisierten Produktionsmaschinen, 2006.

163 Reichle, R.: Einsatz von Internet-Werkzeugen und -Diensten in numerischen Steuerungssystemen für Werkzeugmaschinen, 2006.

164 Kaiser, L.: Systematik für das Qualitäts- und Projektmanagement bei der Abwicklung von Unikatprojekten, 2006.

165 Garber, Th.: Nutzung des redundanten Freiheitsgrades von sechsachsigen Parallelkinematik-Maschinen, 2007.

166 Altenburger, R.: Dynamische Eigenschaften und Regelung mechanisch verkoppelter Antriebssysteme, 2007.

167 Korajda, B.: Steuerungstechnische Verfahren zur echtzeitfähigen Kompensation der Fräserabdrängung, 2007.

168 Röck, S.: Echtzeitsimulation von Produktionsanlagen mit realen Steuerungssystemen, 2007.

169 Priese, J.: Verfahren zur durchgehenden dezentralen Planung in Werkstattstrukturen, 2007.

170 Boye, T.: Vorhersage der kinematischen Kalibriergüte von Parallelkinematiken, 2008.

171 Joannides, M.: Ein Beitrag zur volumenorientierten 3D-Objektrekonstruktion aus digitalen Daten, 2009.

172 Pruschek, P.: Verfahren zur anwendungsgerechten Parametrierung der Steuerung und Regelung von Vorschubachsen, 2009.

173 Fritsch, D.: Steuerung selbstorganisierender Multi-Roboter-Systeme für dynamische Sammelaufgaben am Beispiel der Bekämpfung maritimer Ölverschmutzungen, 2009.

174 Schmitz, S.: Industrielle Powerline-Kommunikation für Antriebseinheiten in Werkzeugmaschinen, 2010.

175 Bengel, M.: Workpiece-centered Approach to Reconfiguration in Manufacturing Engineering, 2010.

176 Weimer, T.: Informationsmodell für die durchgängige Datennutzung in Fabrikplanung und -betrieb, 2010.

177 Oglodin, V.: Maschinenübergreifender agentenbasierter Informationsaustausch für die Störungsbeseitigung, 2010.

178 Kramer, C.: Offene Antriebsreglerplattform, 2011.

179 Stotz, M.: Adaptive Segmentierung von Tiefenbildern für die 3-D-Objektlageerkennung auf Basis von kombinierten regelgeometrischen Elementen, 2011.

180 Selig, A.: Informationsmodell zur funktionalen Typisierung von Automatisierungsgeräten, 2011.