Lehrstuhl für Informationstechnik im Maschinenwesen
der Technischen Universität München

Modellbasierte Generierung einheitlicher Bedienobjekte für verteilte mechatronische Systeme

Heiko G. Meyer

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

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. rer. nat. Heiner Bubb

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Klaus Bender
2. Univ.-Prof. Dr.-Ing. habil. Martin Wollschlaeger,
   Technische Universität Dresden

Die Dissertation wurde am 19. Januar 2004 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 26. März 2004 angenommen.

Modellbasierte Generierung einheitlicher Bedienobjekte für verteilte mechatronische Systeme

In dieser Arbeit wird ein Modell zur standardisierten Beschreibung der Hilfsfunktionalität einheitlicher Bedienobjekte von automatisierungstechnischen Anlagen auf Geräte- und Systemebene entwickelt. Der Gerätehersteller findet durch das Verfahren Unterstützung bei der Realisierung konformer Geräte. Dem Anlagenbauer wird eine Methode an die Hand gegeben, die es ihm ermöglicht, aus den Gerätebeschreibungen die gewünschte Funktionalität des Systems assistentengestützt zu erzeugen und modellbasiert zu beschreiben. Aus den Geräte- und Systembeschreibungen können Software-Komponenten abgeleitet werden, die auf Basis einer etablierten Proxy-Technologie die gewünschte Funktionalität dem Bediener zur Verfügung stellen. Aus Sicht des Anlagenpersonals ist durch die Lösung eine Handhabung von Anlagenteilen äquivalent zu den einzelnen Feldgeräten durch standardisierte Funktionalität und einheitliches Look and Feel gegeben. Zu berücksichtigende Abhängigkeiten der Geräte, beispielsweise bei einer Umprojektierung, notwendige Parametersätze etc. werden vom Systembedienobjekt gekapselt.

Schematic Generation of Unified Operating-Objects for Distributed Mechatronic Systems

In this thesis a standardized reference model to describe the help-functionality of unified operating-objects is developed. The model is valid both on device and system layer. It comprehends the description of uniform functionality, interfaces and the namespace. Vendors of field devices are supported during the realization of compliant devices. Plant engineers and constructors are given a method to generate comfortable unified asset management functionality on system layer. The automatism is based on a model and uses the information stored in standardized descriptions of the underlying devices of a system. The method includes the generation of software components which contain the desired functionality. Unified asset management functionality for systems can be implemented by making use of any proxy technology. Operators of a plant have the advantage of unified tools with similar look and feel and a consistent operating philosophy. Dependent relationships between the devices of a system, e. g. changing the project planning, essential parameters etc., are included by the operating-object of the system.

Vorwort

In den vergangenen fünf Jahren bot sich mir am Lehrstuhl für Informationstechnik im Maschinenwesen der Technischen Universität München die Gelegenheit, mich in hoch interessante Gebiete der Automatisierungstechnik einzuarbeiten. Die Mitarbeit und Leitung an Forschungsprojekten in enger Zusammenarbeit mit den Industriepartnern und das Mitwirken in Industriearbeitskreisen waren wichtige Voraussetzungen, um mir das entsprechende Fachwissen aneignen zu können.

An erster Stelle möchte ich mich bei meinen Eltern bedanken, die mir den Weg zu dieser Arbeit in jeder Hinsicht überhaupt erst ermöglichten. Bei meiner Ehefrau und meinen Kindern möchte ich mich für die Geduld und das mir immer zu entgegengebrachte Verständnis bedanken.

Insbesonders gebührt Dank meinem Doktorvater Herrn Professor Dr. Bender, der mir stets den notwendigen Freiraum zur Entfaltung und Erprobung meiner eigenen Ideen gelassen hat. Sein Vertrauen in meine Handlungsweise bei der Leitung eines großen Verbundprojektes hat über die Forschungsarbeit hinaus wesentlich zur Erreichung meiner mir gesetzten Ziele beigetragen.

Ganz herzlich bedanke ich mich bei Herrn Professor Dr. Wollschlaeger für das Interesse an meiner Arbeit und die Übernahme des Koreferats. Herrn Professor Dr. Bubb möchte ich meinen Dank für den Prüfungsvorsitz aussprechen.

Besonderen Dank gebührt meinem Kollegen Herrn Kuttig, der mir sowohl beim Tagesgeschäft am Institut als auch beim Schreiben meiner Dissertation stets zur Seite stand und durch zahlreiche wissenschaftliche Diskussionen mir viele wertvolle Anregungen für meine Arbeit gegeben hat. Allen Mitarbeitern am Lehrstuhl möchte ich Danken für die gute Zusammenarbeit.

Gerade die vielen prototypischen Realisierungen und Softwaretools die während der Projekte entstanden sind, waren nur machbar durch die tatkräftige Unterstützung interessierter Studenten, die sowohl in Form von diversen Diplom- und Semesterarbeiten als auch über längeren Zeitraum als wissenschaftliche Hilfskraft zum Erreichen der Projektziele nicht unerheblich beigetragen haben. An dieser Stelle möchte ich mich für das außerordentlich große Engagement bei der Verwirklichung der Ideen und Konzepte bei allen beteiligten Studenten bedanken.

München, im Dezember 2003 Heiko Meyer

Inhaltsverzeichnis

Glossar

Anmerkung

Siehe hierzu auch das Stichwortverzeichnis. Querverweise auf weitere Begriffe des Glossars sind durch „→“ gekennzeichnet.

Anlagen-Engineering

Summe aller Maßnahmen zur Inbetriebnahme, Instandhaltung, Instandsetzung und Wartung einer automatisierungstechnischen Anlage.

Architektur

Eine Architektur beschreibt die Elemente und ihre Beziehung zueinander um eine bestimmte Funktion zu erfüllen. Referenzarchitekturen beschreiben eine schematische Struktur, die für eine bestimmte Anwendung konkretisiert werden muss.

Bedienobjekt

Software-Objekt, das die gesamten →Hilfsfunktionen eines Feldgerätes oder →mechatronischen Systems über eine Benutzerschnittstelle dem Anwender zur Verfügung stellt.

Black-box Integration

Integrationsansatz, bei dem die integrierten Komponenten als Ganzes verwendet werden, ohne ihre innere Feinstruktur zu kennen.

Client-Server Modell

Kooperationsmodell, bei dem eine Instanz eine Dienstleistung einer anderen Instanz anfordert.

COM

Component Object Model. Spezifikation der Firma Microsoft zur Umsetzung von Software-Komponententechnologien auf Basis der Windows-Betriebssysteme.

DCOM

Distributed COM. Erweiterung von COM für die Kommunikation verteilter Komponenten in lokalen und globalen Netzen.

DDD

DeKOS Device Description. →DeKOS-konforme, →XML-basierende Beschreibung der →Hilfsfunktionalität eines Feldgerätes oder Systems.

DeKOS

Dezentrale Kooperierende Offene Mikrosysteme. Verbundprojekt des Bundesministeriums für Bildung und Forschung (Förderkennzeichen: 16SV894/1).

DRA

Formale Beschreibung des →DeKOS-Bedienmodells in Form eines →XML-Schemas.

DTM

Device Type Manager. →Proxy zur Abbildung der Funktionalität eines Feldgerätes, das eingebettet in einer speziellen Rahmenapplikation, der so genannten →FDT Frameapplication, die Bedienung des Gerätes ermöglicht.

DTD

Document Type Definition. Beschreibt die Struktur eines XML-Dokumentes; weitestgehend abgelöst durch →XML-Schemata.

Engineering

→Anlagen-Engineering, →Entwicklungsprozess

Engineering-Prozess

→Entwicklungsprozess

Entwicklungsprozess

Eine bestimmte Menge an Aufgaben und Aktivitäten, die mit einer übergeordneten Zielsetzung in einer definierten Reihenfolge (sequentiell und/oder parallel) bearbeitet werden. Im Zusammenhang der Arbeit das Vorgehensmodell zur Generierung eines Bedienobjektes.

FDT

Field Device Tool. Spezifikation eines Schnittstellenkonzeptes, das den universellen Einsatz von →DTMs in unterschiedlichen Applikationen beliebiger Hersteller vorsieht.

Hilfsfunktionen

Die gesamte Funktionalität eines Feldgerätes zur Projektierung und zum Service, die dazu dient, das Gerät in den Zustand zu bringen, in dem die →Prozessfunktion im Betrieb optimal genutzt werden kann.

itm

Lehrstuhl für Informationstechnik im Maschinenwesen der Technischen Universität München.

Mechatronisches System

System, bei dem die zu erbringende Funktionalität durch eine enge Verknüpfung mechanischer, elektronischer und informationstechnischer Bestandteile erzielt wird. Eine automatisierungstechnische Anlage ist in Anlehnung an die Konstruktion in mehrere Systeme unterteilbar.

Modell

Eine formalisierte Abbildung der Wirklichkeit, beschränkt auf die Elemente, die für eine bestimmte Zielsetzung notwendig sind.

Paradigma

Eine bestimmte Art und Weise, die Problemwelt zu begreifen und Lösungskonzepte zu formulieren.

Partialsystem

→Mechatronisches System

PNO

Profibus Nutzerorganisation e. V.

Proxy

Ein Software-Objekt, das als Stellvertreter eines anderen, in der Regel nicht lokalen Objektes fungiert. Der Proxy übernimmt, für den Anwender transparent, die Rolle des originalen Objektes.

Prozessfunktion

Automatisierungsfunktion eines Feldgerätes zur Lösung der für das Gerät festgelegten Automatisierungsaufgabe.

Referenzarchitektur

→ Architektur

Schnittstelle

Zugang zu Informationen und/oder Funktionalität eines Systems. Unterscheiden lassen sich proprietäre Schnittstellen, deren Spezifikation nur dem Entwickler bekannt ist und offene Schnittstellen, deren Spezifikation allgemein verfügbar ist.

White-box Integration

Ein Integrationsansatz, bei dem die zu integrierenden Komponenten in ihrer Feinstruktur bekannt sind und dies für die Integration essentiell notwendig ist und somit auch genutzt wird.

XML

EXtensible Markup Language. Eine universelle Beschreibungssprache zur textuellen Notation von Daten und Datenmodellen.

XML-Schema

Beschreibt ähnlich zur →DTD die Struktur eines XML-Dokumentes.

1 Einleitung

1.1 Motivation

Ein wichtiger Antrieb für Innovationen in der Güter produzierenden Industrie ist der Bedarf nach Steigerung der Produktivität. Dies betrifft primär das Ziel der Optimierung des Herstellungsprozesses. Sekundär beinhaltet dies aber auch das Ziel der Minimierung des gesamten Engineering-Aufwands für Produktionsanlagen von der Produktidee bis hin zum Recycling, da dieser einen enormen Aufwandsanteil darstellt.

Die Engineering-Kosten automatisierter Großanlagen betragen derzeit typischerweise ca. 60% der gesamten Investitionskosten [PNO 02A]. Der Anspruch des Anlagenbetreibers nach immer höherer Produktivität, Qualität und Flexibilität führt zwangsläufig zu immer komplexeren Automatisierungslösungen. Waren Innovationen der letzten Jahrzehnte noch durch die Steigerung des Automatisierungsgrades geprägt, liegt das Optimierungspotenzial heute größtenteils in der Reduzierung des Engineering-Aufwands.

Die Betreiber von automatisierungstechnischen Anlagen werden bei der Bedienung[1] dieser mechatronischen Systeme (makroskopisch betrachtet), bestehend aus Teilanlagen, vor eine kaum beherrschbare Aufgabe gestellt. Die vorhandene Funktionsvielfalt der eingesetzten Geräte, ermöglicht durch die Zunahme der Leistungsfähigkeit, zieht eine erhöhte Komplexität hinsichtlich der Bedienung nach sich. Zur Kompensation des Defizits liefert der jeweilige Gerätehersteller in der Regel ein eigenes Bedientool für sein Gerät mit. Der Bediener muss folglich mit Tools arbeiten, die größtenteils auf völlig unterschiedlichen Bedienphilosophien basieren. Um sich nicht durch die „Hintertür“ ein weiteres neues Bedienkonzept mit allen Folgen wie Schulungen und Akzeptanzproblemen einzukaufen, verwenden größere Anwenderfirmen als Konsequenz möglichst nur Geräte bestimmter Hersteller, auch wenn am Markt bessere oder günstigere Anbieter existieren.

Mit dem Wunsch des Betreibers nach einer einheitlichen Bedienung intelligenter Feldgeräte beschäftigte sich u. a. das Verbundprojekt DeKOS[2]. Als Ergebnis ist,

---

[1] Projektierung und Service von intelligenten Feldgeräten

[2] Dezentrale Kooperierende Offene Mikrosysteme, Verbundprojekt des Bundesministeriums für Bildung und Forschung (Förderkennzeichen: 16SV894/1)

auf Basis einer standardisierten Beschreibung der Gerätefunktionalität für die Projektierung und zum Service (Hilfsfunktionalität), ein einheitliches Bedienmodell für Feldgeräte aller Geräteklassen entstanden [BENDER ET AL. 02]. Jedoch wurde immer noch nicht die Herausforderung gelöst, dass weiterhin für jedes Gerät ein eigenes Bedientool vorliegt, auch wenn die Geräte im Verbund eine Gesamtfunktionalität erfüllen, die für den Bediener maßgebend ist. Bei größeren Anlagen ist die hohe Anzahl an Tools als Problem geblieben. Gravierender Nachteil ist, dass dem Anlagenpersonal die Logik beim Bedienen des Systems bekannt sein muss. Dies beinhaltet u. a. die Abhängigkeiten der einzelnen Geräte wie Übersetzungsverhältnisse, Rampen etc., die sich aus der Konstruktion ergeben.

Ansätze wie die des Asset Managements bringen durch eine Kapselung der Hilfsfunktionalität von Anlagenteilen Verbesserungen in Bezug auf die Diagnose der Gesamtanlage; allerdings sind die bestehenden Lösungen eingeschränkt auf die Domäne der Verfahrenstechnik und die Phase der Instandhaltung. Die Projektierung der einzelnen Feldgeräte wird durch das Asset Management nicht gesondert unterstützt, sodass diese mit klassischen Methoden zu erfolgen hat. Dem Mehrwert durch erweiterte Diagnosemöglichkeiten steht der zusätzliche Aufwand bei der Handhabung der benötigten speziellen Feldgeräte gegenüber. Des Weiteren existiert kein einheitliches Modell und keine einheitliche Vorgehensweise zur Realisierung der zusätzlichen Hilfsfunktionalität; es muss auf propritäre Lösungen zurückgegriffen werden. Dies hat wiederum einen nicht unerheblichen erweiterten Engineering-Aufwand zur Folge. Es ist keine einheitliche Lösung für Geräte und Systeme vorhanden, die den Anforderungen der Anlagenbetreiber gerecht wird.

1.2 Ziel der Arbeit

Das im vorhergehenden Abschnitt beschriebene Problem ist durch eine geeignete Methode zur standardisierten Kapselung der Hilfsfunktionalität von Teilanlagen zu lösen. Dabei sind sowohl die Anforderungen des Endanwenders wie die des Anlagenbauers und des Herstellers der intelligenten Feldgeräte zu berücksichtigen. Die Basis des hier verfolgten Lösungsansatzes bildet die modellbasierte Beschreibung der Hilfsfunktionalität von Teilanlagen. Das durchgängige Engineering-Konzept vom einzelnen Feldgerät bis hin zur Teilanlage ermöglicht eine komfortable Generierung der Softwarekomponente zur Bedienung von Teilanlagen; dabei unterstützt die Lösung den gesamten Engineering-Prozess. Die sich hieraus unmittelbar ergebenden Vorteile sind nachfolgend aufgeführt:

Der Bediener einer automatisierungstechnischen Anlage kann Teile einer Anlage genauso wie einzelne Feldgeräte zentral und komfortabel ohne Kenntnis der Interna handhaben. Dabei ist die zu bedienende Teilanlage aus seiner Sicht eine Black-Box, d. h. zu berücksichtigende Abhängigkeiten der Geräte bleiben ihm verborgen. Diese sind in der Software zur Bedienung der Teilanlage realisiert. Die Funktionalität des Systems wird standardisiert angeboten, wodurch eine intuitive Bedienung gegeben ist. Dies beinhaltet die Existenz einer einheitlichen Bedienphilosophie ebenso wie eine einheitliche Semantik der angebotenen Hilfsfunktionen und das Look&Feel. Somit minimiert sich seitens des Anlagenbetreibers der Schulungsaufwand für das Personal erheblich. Zudem sind weniger Spezialisten zum Anlagenerhalt notwendig, da das Spezialwissen über die Interna der Anlagenteile in der Bediensoftware integriert ist.

Zunächst ist der Anlagenbauer gefordert, dem Wunsch des Auftraggebers bezüglich der einfachen Handhabung von Teilanlagen nachzukommen und bei der Realisierung zu berücksichtigen. Die vorliegende Arbeit beschreibt ein Verfahren, mit dessen Hilfe eine weitestgehend automatische Generierung der Bedienelemente von Teilanlagen ohne großen Aufwand gegeben ist. Dies kommt dem Anlagenbauer auch selbst zugute, da die Kapselung eine raschere Inbetriebnahme gewährleistet. Die Wiederverwendung von Software-Komponenten senkt den Engineering-Aufwand zusätzlich und steigert die Konkurrenzfähigkeit. Der Aspekt des Asset Managements findet im spezifizierten Modell Unterstützung, sodass eine einfache Realisierung zusätzlicher Diagnosefunktionalität gegeben ist.

Das in der Arbeit behandelte Modell definiert einen minimalen Umfang an Hilfsfunktionen, über die jedes intelligente Feldgerät idealerweise verfügt. Der Gerätehersteller kann die benötigten standardisierten Hilfsfunktionen zur Beschreibung seines Feldgerätes in strukturierter Form auswählen. Hierdurch ist eine aufwandsarme Erstellung der konformen Gerätebeschreibungen gegeben. Dies ist die Basis für den Anlagenbauer, eine einfache Realisierung der Bedienelemente von Teilanlagen ohne großen Mehraufwand durchführen zu können.

Abschließend erfolgt die Umsetzung des spezifizierten Modells durch eine prototypische Realisierung. Der vollständige Engineering-Prozess zur Erstellung des Bedienobjektes eines Beispielsystems aus der Verfahrenstechnik wird an dieser Stelle ausführlich betrachtet. Die prototypische Realisierung wird u. a. bezüglich des Engineering-Verfahrens bewertet und gegen die Anforderungen validiert.

1.3 Gliederung der Arbeit

Der Aufbau der vorliegenden Arbeit ist schematisch in Abbildung 1-1 dargestellt. Nach dem einleitenden Kapitel werden in Kapitel 2 „Anforderungsanalyse“ zunächst die wichtigsten, häufig benutzen Begriffe eingeordnet und definiert. Nach einer Beschreibung des allgemeinen Engineering-Lifecycles offener Automatisierungssysteme folgt detailliert die Beschreibung der Engineering-Aufgaben von Teilanlagen. Die Defizite durch die zu berücksichtigenden Abhängigkeiten und das notwendige Wissen aus der Konstruktion werden aufgeführt und abschließend die Aufgabenstellung formuliert.

Das Kapitel 3 „Engineering-Konzepte und Technologien“ zeigt, dass die in Kapitel 2 erarbeiteten Defizite mit herkömmlichen Mitteln der Automatisierungstechnik nicht kompensierbar sind. Dementsprechend wird ein geeigneter Lösungsansatz für das identifizierte Problem vorgestellt.

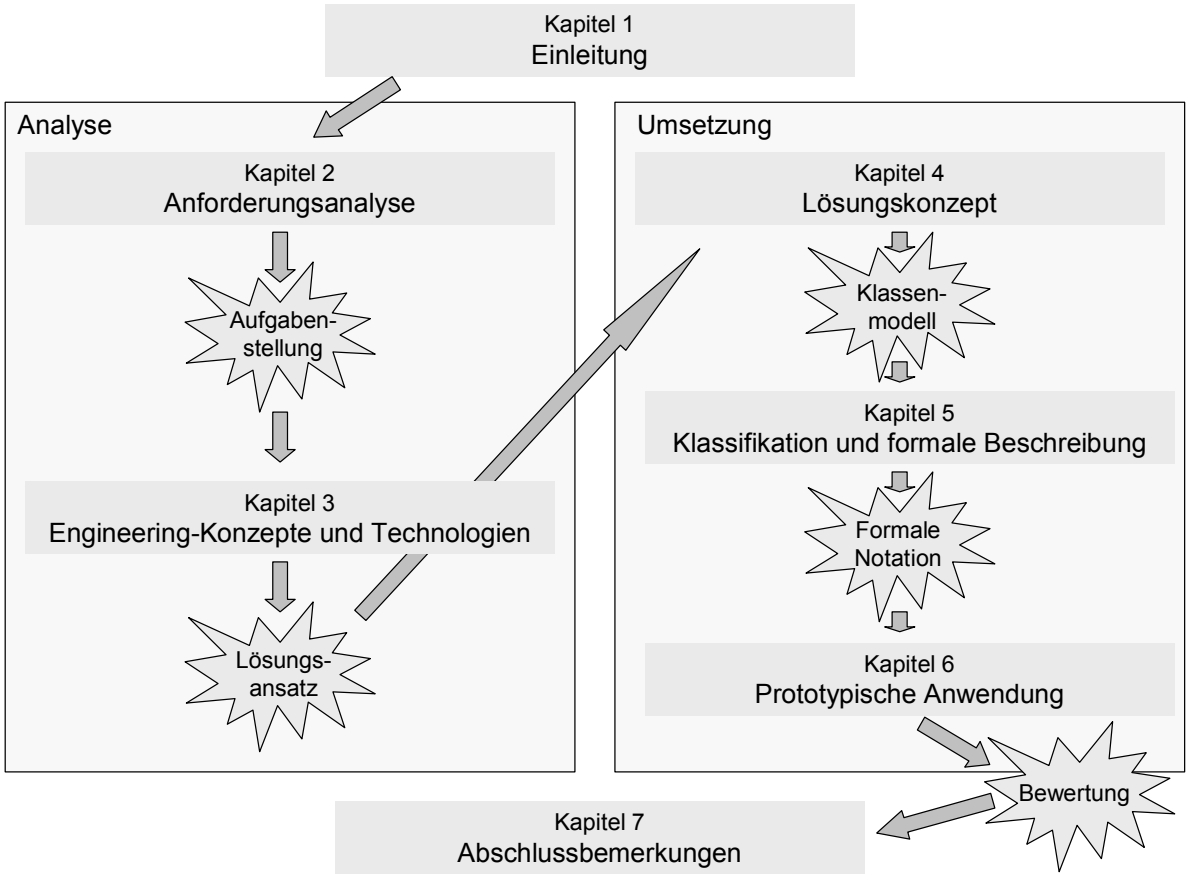


Abbildung 1-1: Aufbau der Arbeit

In Kapitel 4 „Lösungskonzept“ wird dieser Lösungsansatz ausgearbeitet. Hierzu wird näher auf die Engineering-Aufgaben von mechatronischen Systemen im gesamten Lifecycle eingegangen. Nach dem Beleg des Bedarfs nach Modellierung folgt die Herleitung der Voraussetzungen zur Generierung einheitlicher Bedienobjekte für Teilanlagen. Kern des Modells ist neben der Abbildung der Geräteabhängigkeiten einer Teilanlage ein Klassifikationsansatz der standardisierten Hilfsfunktionalität intelligenter Automatisierungskomponenten.

Die Ausarbeitung des Lösungsansatzes beschreibt Kapitel 5 „Klassifikation und formale Beschreibung“; die Definition spezieller Systemgrundfunktionen und deren Eigenschaften erfolgt. Zur Beschreibung der funktionalen Abhängigkeiten zwischen den Geräten einer Teilanlage wird eine Referenzarchitektur spezifiziert. Zusammen mit der klassifizierten Funktionsbibliothek folgt abschließend die formale Beschreibung der zuvor erstellten Modelle.

Kapitel 6 „Prototypische Anwendung des Modellierungskonzeptes“ beschäftigt sich mit der Erprobung der klassifizierten Funktionsbibliothek und der zuvor spezifizierten Modelle. Die Erstellung von zum Modell konformen Gerätebeschreibungen und die modellbasierte Generierung einheitlicher Bedienobjekte für Teilanlagen auf Basis der zuvor erstellten Gerätebeschreibungen wird demonstriert. Im weiteren Verlauf wird anhand geeigneter Technologien die technische Umsetzung einer derartigen Lösung prototypisch beschrieben.

Abschließend erfolgt im Kapitel 7 „Abschlussbemerkungen“ neben einer Zusammenfassung der Ergebnisse eine Bewertung der Arbeit und eine Validierung des Lösungskonzeptes gegenüber der Aufgabenstellung. Die Möglichkeiten zukünftiger weiterführender Arbeiten werden in einem Ausblick diskutiert.

2 Anforderungsanalyse

Zur Erbringung der heute geforderten Aufgaben bezüglich der Produktion und der daraus resultierenden Produkte kommen in modernen Anlagen in der Regel eine große Anzahl verschiedener Feldgerätetypen, zum Teil auch unterschiedliche Feldbusse zum Einsatz. Die Inbetriebnahme der Anlagen gestaltet sich damit schwierig und zeitintensiv [SIMON 03]. Die zu lösenden neuen Herausforderungen sind nachfolgend zu diskutieren.

Es werden zunächst Begriffe, die für die Arbeit relevant sind, eingeführt. Anschließend wird auf die notwendigen Engineering-Aufgaben im Lifecycle moderner Automatisierungssysteme Als Bestandteil von Produktionsanlagen näher eingegangen. Dabei findet ausgehend von den allgemeinen Aufgaben eine Eingrenzung auf den betrachteten Bereich der Inbetriebnahme- und Instandhaltungsmaßnahmen von Teilanlagen statt. Nach einer Analyse der Defizite bei der Bedienung komplexer Produktionsanlagen ist abschließend die Aufgabenstellung anzuleiten.

2.1 Begriffsbildung

2.1.1 Geräteverbund

Nachfolgend werden die wichtigsten Fachbegriffe, die in der Arbeit verwendet werden, eingeführt. Begriffe, die in verschiedenen Zusammenhängen in der Literatur Verwendung finden, werden kontextbezogen definiert.

Definition 2.1

Ein **intelligentes Feldgerät** (Sensor zur Messwerterfassung, Aktor zur gezielten Beeinflussung des Prozesses) ist eine Komponente[3] der Automatisierungseinrichtung, die über eine Kommunikationsanschaltung (Feldbus oder Netzwerkanschluss[4]) und ausreichend Rechenkapazität für Vorverarbeitungs- bzw. Zusatzfunktionen verfügt [BENDER 02].

Die folgenden Kapitel beschäftigen sich ausschließlich mit intelligenten Feldgeräten als essentieller Bestandteil von modernen Automatisierungssytemen. Dabei spielt es aus Sicht des Modells zur Handhabung der Geräte keine Rolle, ob die Rechenkapazität in Form eines Microcontrollers direkt im Feldgerät oder extern, z. B. PC-basiert, realisiert ist.

Neben der Betrachtungsweise einer Anlage auf Geräteebene, bei der jedes Feldgerät, wie beispielsweise eine Ventilinsel, für sich eigenständig zu sehen ist, wreden zur Erbringung der geforderten Funktion mehrere Feldgeräte auch zu einem Geräteverbund zusammengefasst. In der Literatur sind verschiedene Definitionen für solche Zusammenschlüsse mindestens zweier Geräte zu finden:

Definition 2.2

Ein **Subsystem** ist ein Verbund mehrerer Feldgeräte, die innerhalb einer Anlage eine logische Einheit darstellen. Diese erfüllt eine Automatisierungsaufgabe, festgelegt durch den geforderten Prozess, der essentieller Bestandteil der Einheit ist. Ein Subsystem besitzt eine klar definierte Schnittstelle nach außen sowie eine festgelegte Funktionalität. Es erfüllt alle Anforderungen einer Black-Box[5] und weist hierdurch dieselben Vorteile auf.

Der Begriff „Sub“ impliziert im deutschen Sprachgebrauch, dass es sich um eine hierarchische Anlagenstruktur handelt. Dementsprechend muss ein Subsystem mindestens ein übergeordnetes System, ein so genanntes Supersystem aufweisen [RIEGER 95].

---

[3] Im Gegensatz zur Software-Komponente findet der Begriff in diesem Kontext Verwendung für den gerätetechnischen Bestandteil einer Produktionsanlage.

[4] Ethernet-Anschaltung

[5] Komponente bestehend aus Hard- und Software mit festgelegter Funktionalität. Wie die Funktionalität dabei erbracht wird, ist vollkommen irrelevant. Jede Black-Box kann durch jede andere Black-Box mit identischen Eigenschaften und Schnittstellen ersetzt werden [VDI 3633].

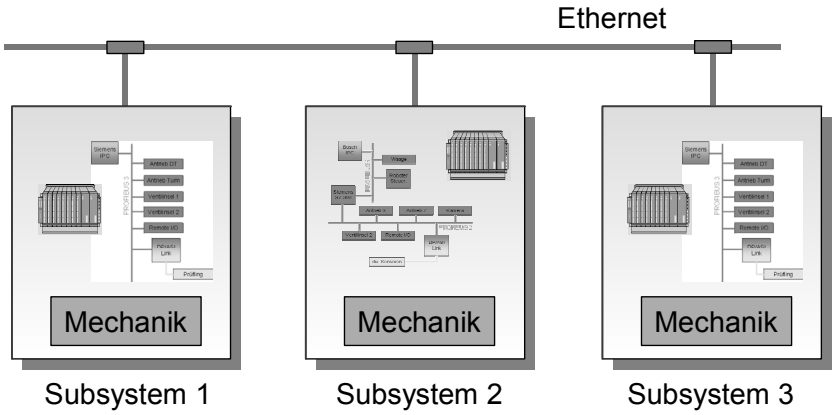


Abbildung 2-1: Kommunikation von 3 Subsystemen bei PROFInet

Entgegen dieser Betrachtungsweise wird bei Komponenten-basierten Technologien wie der PROFInet-Technologie[6] bei den dort betrachteten Geräteverbünden ebenfalls der Begriff Subsystem im Sinne des englischen Wortes „Teil“ verwendet. Wie in Abbildung 2-1 zu sehen ist, können dabei auch drei gleichberechtigte Subsysteme ein Gesamtsystem bilden. Dementsprechend werden in der vorliegenden Arbeit Zusammenschlüsse von Feldgeräten mit nachfolgenden Eigenschaften betrachtet:

Definition 2.3

Ein **Partialsystem** ist ein **verteiltes mechatronisches System** bei dem der Systemzweck[7] durch eine enge Verknüpfung mechanischer, elektronischer und informationstechnischer Bestandteile zu erzielen ist. Das System ist aus Sicht der zu erfüllenden Funktion nicht teilbar. Der Prozess ist substantieller Bestandteil des Systems.

Eine Produktionsanlage setzt sich somit aus mehreren mechatronischen Systemen zusammen. Über die Art und Weise, wie die Partialsysteme miteinander kommunizieren, ist dabei nichts ausgesagt. So können beispielsweise zwei Systeme identische Funktionen zur Inbetriebnahme aufweisen, obwohl sie über verschiedene Kommunikationsstandards nach außen verfügen. So handelt es sich bei einem in dieser Arbeit betrachteten mechatronischen System nicht um eine Black-Box.

[6] Herstellerübergreifender Standard zur Anlagen-weiten Integration von verteilten Automatisierungssystemen auf Basis von Ethernet [PNO 02A].

[7] Die zu erbringende Funktionalität des Systems.

2.1.2 Bedienfunktionalität

Unter Automatisierung versteht man im Allgemeinen, Mittel einzusetzen, um einen Vorgang in einem technischen System selbsttätig ablaufen zu lassen [DIN 19233]. Dabei wird durch den Automaten[8] eine gezielte Prozessbeeinflussung sichergestellt.

Definition 2.4

Ein **Prozess** ist die Gesamtheit von aufeinander einwirkenden Vorgängen in einem System, durch die Materie, Energie oder Information umgeformt, transportiert oder gespeichert wird. Ein **technischer Prozess** ist ein Prozess, dessen physikalische Größen mit technischen Mitteln erfasst und beeinflusst werden [DIN 66201].

Nach der Art des zu automatisierenden technischen Prozesses wird unterschieden zwischen der Automatisierung eines

- Verfahrensprozesses und eines
- Fertigungsprozesses.

Dementsprechend wird die Unterscheidung oft auch in Verfahrenstechnik und Fertigungstechnik vorgenommen. Irreführend hingegen sind die Bezeichnungen Prozess- und Fabrikautomatisierung, da gerade in der englischsprachigen Literatur der Begriff „process automation“ oft als Synonym für den Begriff „industrial automation“ Verwendung findet [LAUBER/GÖHNER 99A].

In der Verfahrenstechnik wird zudem zwischen dem Leiten[9] des Prozesses und der Bedienung der in der Anlage befindlichen Geräte unterschieden [POLKE 92]. In der Fertigungstechnik ist dieses Unterscheidungsmerkmal nicht gebräuchlich, da bei diesen Anlagen zur Laufzeit selten eine manuelle Prozessbeeinflussung vorgenommen wird.

Definition 2.5

Das **Leiten** ist die Gesamtheit aller Maßnahmen, die einen im Sinne festgelegter Ziele erwünschten Ablauf eines Prozesses bewirken. Die Maßnahmen werden vorwiegend unter Mitwirkung des Menschen aufgrund der aus dem Prozess oder aus der Umgebung erhaltenen Daten mit Hilfe der Leiteinrichtung getroffen [DIN 19222].

---

[8] Künstliches System, das selbsttätig ein Programm befolgt [DIN 19233].

[9] Das Leiten eines Prozesses wird oft auch als Prozessführung oder Prozessbedienung bezeichnet.

Die Daten aus dem Prozess werden gewöhnlich zyklisch von den Sensoren der Steuerung zur Verfügung gestellt. Das gezielte Einwirken auf den Prozess geschieht durch die Aktoren. Somit erfüllt jedes Automatisierungsgerät eine bestimmte Funktionalität zur Zielerreichung des Prozesses [SCHNIEDER 99]. Diese wird über eine bestimmte Automatisierungsaufgabe festgelegt.

Die Klassifikation der Funktionalität einer allgemeinen Automatisierungskomponente ist Abbildung 2-2 zu entnehmen. Neben der Kernfunktionalität, die das Gerät im Betrieb zu erfüllen hat, gibt es eine Vielzahl von weiteren Funktionen, die für den ordnungsgemäßen und zielerreichenden Betrieb einer Anlage notwendig sind.

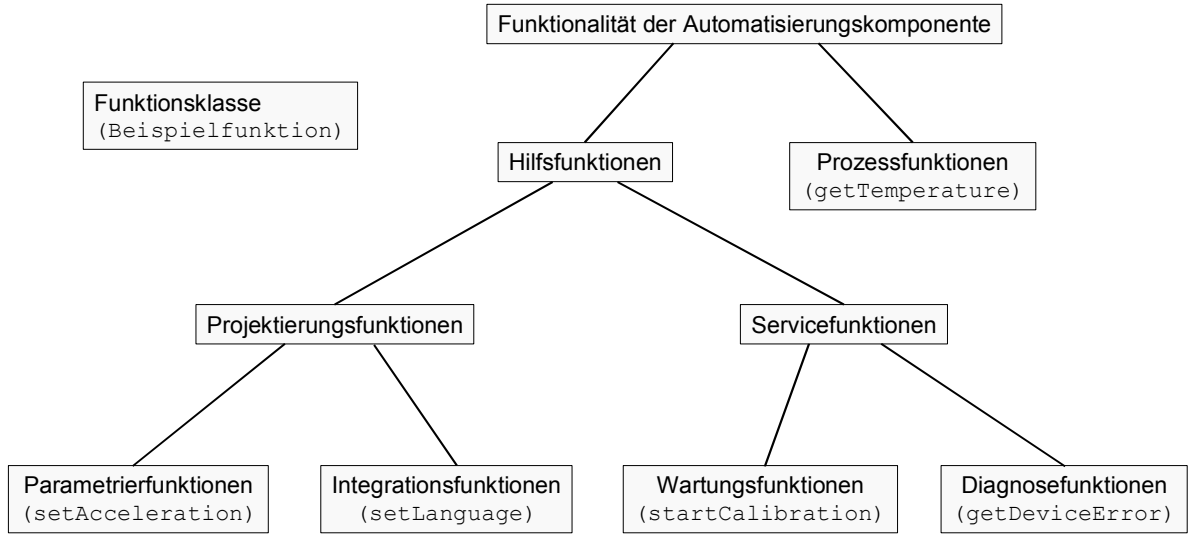


Abbildung 2-2: Funktionalität der Automatisierungskomponente

Definition 2.6

Die **Prozessfunktion** eines Feldgeräts ist die Automatisierungsfunktion zur Lösung der für das Gerät festgelegten Automatisierungsaufgabe [BREGULLA 01].

Neben der Prozessbedienung spielt die Gerätebedienung eine wesentliche Rolle bei der Handhabung von modernen Produktionsanlagen. Neben der Prozessfunktion, d. h. der eigentlichen Funktionalität, die das Feldgerät im Betrieb zur Verfügung stellen muss, bieten intelligente Feldgeräte eine Vielzahl von zusätzlichen Funktionen an.

Definition 2.7

Die **Hilfsfunktionen** eines Feldgeräts stellen die gesamte Funktionalität zur Projektierung und zum Service des Geräts zur Verfügung, die dazu dient, das Gerät in den Zustand zu bringen, in dem die Prozessfunktion im Betrieb optimal genutzt werden kann.

Dabei übersteigt die Menge der Hilfsfunktionen die Menge der Prozessfunktionen, meist nur eine pro Gerät, bei weitem. Spezielles Know-how eines Herstellers liegt in der Regel ohnehin in der Prozessfunktion verborgen, z. B. wie schnell und präzise ein Messwert zur Verfügung steht, während die Funktionalität zur Bedienung eines Geräts von jedem Hersteller gleichermaßen zur Verfügung gestellt werden muss [BENDER ET AL. 02].

Oft wird der Vorgang des Einstellens eines Gerätes auch als Parametrierung bezeichnet. Dies liegt darin begründet, dass bei den meisten Feldgeräten eine Hilfsfunktion über Schreiben und Lesen spezieller Parameter im Gerät erfolgt.

2.1.3 Software-Technik

Durch die enorme Funktionsvielfalt der intelligenten Feldgeräte sind diese innerhalb der Geräteklasse universell einsetzbar. Bei der Inbetriebnahme der Geräte werden diese dann über die zur Verfügung gestellte Hilfsfunktionalität dem Einsatzzweck entsprechend angepasst. Bei einfachen Geräten, wie beispielsweise einem Drucktransmitter, kann dies durch Schreiben und Lesen spezieller Geräteparameter über den zyklischen Datenverkehr[10] durch das Steuerungsprogramm erfolgen. Bei komplexeren Feldgeräten, wie Frequenzumrichtern oder Messgeräten im Bereich der Verfahrenstechnik, erfolgt die Projektierung meistens über spezielle Softwaretools[11], die vom Hersteller mit dem Gerät ausgeliefert werden und meistens gesonderte Kommunikationswege, wie beispielsweise eine serielle Schnittstelle verwenden.

Definition 2.8

Ein **Bedientool** ist ein eigenständiges Anwendungsprogramm, das die gesamte Hilfsfunktionalität eines intelligenten Feldgerätes dem Bediener[12] PC-basierend, über eine entsprechende Benutzerschnittstelle, zur Verfügung stellt.

---

[10] Die Prozessfunktion eines Feldgerätes wird realisiert über das zyklische Schreiben und Lesen von Daten auf dem Feldbussystem. Zyklische Dienste wiederholen sich dabei in einer definierten Zeit [EN 50170].

[11] Anwendungsprogramme, auch als Parametriertool bezeichnet.

[12] Anlagenbediener im Sinne des Rollenmodells nach [VDI 2187].

Die Vielzahl der heutigen Softwaretools stellen den Benutzerdialog über eine graphische Benutzerschnittstelle[13] dar. In der Verfahrenstechnik etablieren sich zunehmend Bussysteme wie Profibus-PA[14], die neben dem zyklischen Datenverkehr azyklische Dienste[15] zur Projektierung und zum Service der Geräte zur Verfügung stellen [RÖMER 02]. Hierdurch ist eine Minimierung des Aufwands für die Realisierung der Kommunikation zwischen Feldgerät und Bedientool gegeben. Je nach Ausführung des Tools ist die Bedienung einzelner oder auch mehrerer verschiedene Feldgeräte gegeben.

Aus unterschiedlichsten Gründen ist es nicht immer möglich oder auch sinnvoll, auf bestimmte Geräte direkt zuzugreifen. Insbesondere gilt dies für Software-Objekte[16] von Geräten, die innerhalb anderer Rechnerprozesse oder sogar Rechnergrenzen laufen. Ein Beispiel hierfür aus dem Bereich der Büro-Informationstechnik ist die Anbindung eines beliebigen Druckers an ein Anwendungsprogramm bei einem Windows[17]-basierten Betriebssystem. Passend zum Drucker wird betriebsystemseitig ein virtuelles Gerät, der Druckertreiber, installiert. Dieser bildet die gesamte Funktionalität des realen Gerätes ab und stellt sie anderen Applikationen zur Verfügung. Das Textverarbeitungsprogramm schickt einen Druckauftrag nicht direkt zum Gerät, sondern druckt das gewünschte Dokument auf dem virtuellen Gerät. Die Kommunikation zum Gerät und Abarbeitung des Auftrags wird durch den Gerätetreiber sichergestellt. Die Funktionalität des Gerätes steht allen Anwendungsprogrammen gleichermaßen zur Verfügung (s. Abbildung 2-3). Diese Architektur löst man softwareseitig mit Hilfe eines Stellvertreterobjekts, einem so genannten Proxy [HOANG/RIEGER 99].

---

[13] So genanntes GUI (Graphical User Interface).

[14] Profibus-Version für die Prozess Automation.

[15] Azyklische Dienste sind Kommunikationstelegramme, deren Auftreten von einmaligem Charakter ist und die zeitliche Abfolge der zyklischen Dienste nicht behindern [EN 50170].

[16] In der objektorientierten Softwareentwicklung besitzt ein Objekt bestimmte Eigenschaften und reagiert mit einem definierten Verhalten auf seine Umgebung. Es besitzt eine Identität, die es von allen anderen Objekten unterscheidet. Die Eigenschaften eines Objekts werden durch dessen Attributwerte ausgedrückt, sein Verhalten durch eine Menge von Funktionen [BALZERT 95].

[17] registriertes Warenzeichen der Microsoft-Corporation

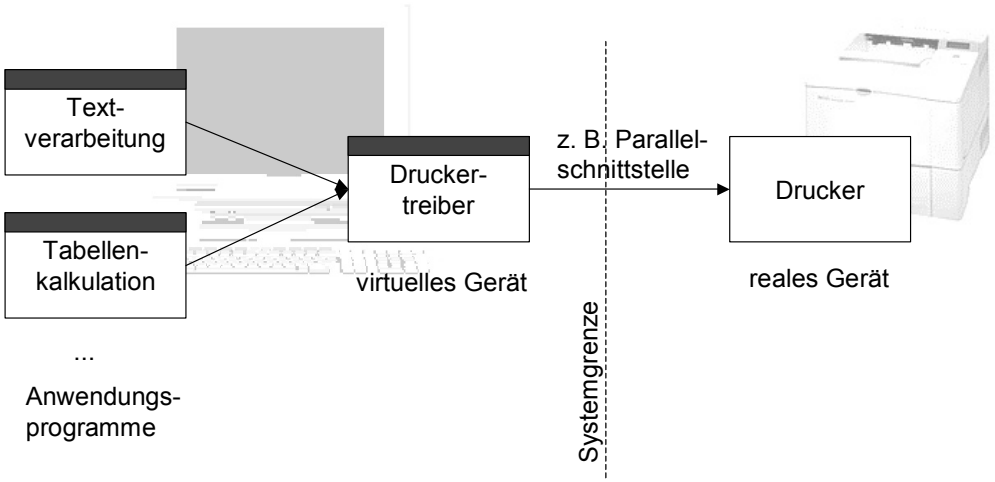


Abbildung 2-3: Druck aus Anwendungsprogramm über Druckertreiber

Definition 2.9

Ein **Bedienobjekt** ist ein Software-Objekt im Sinne eines Proxys, das die gesamte Hilfsfunktionalität eines intelligenten Feldgerätes dem Anwender über eine integrierte Benutzerschnittstelle und zusätzlich anderen Software-Objekten über eine öffentliche Schnittstelle[18] zur Verfügung stellt.

Die meisten Bedienproxys basieren auf Technologien, bei denen das Objekt nur in einem Rahmenanwendungsprogramm[19] lauffähig ist. Eine Proxy-Technologie, die sich in der Automatisierungstechnik zunehmend etabliert, ist FDT (Field Device Tool) [MEYER 01]. Dabei wird die Funktionalität ausschließlich dem Benutzer über eine integrierte Benutzerschnittstelle und nicht zusätzlich über eine öffentliche Schnittstelle angeboten. Diese Eigenschaft ist aber bei den betrachteten Bedienproxys der vorliegenden Arbeit essentiell notwendig. Die Gesamtheit aus einer Frameapplication und den eingebetteten Bedienproxys bildet ein Bedientool.

[18] auch als Public Interface bezeichnet
[19] auch als Frame Application bezeichnet

2.2 Engineering im Lifecycle offener Automatisierungssysteme

2.2.1 Die offene Systemarchitektur

Bei offenen Systemen wird der Grad der Offenheit abgestuft. Dabei versteht man nach EN 50170 unter:

- **Interconnectivity**, dass der Austausch von Daten zwischen den einzelnen Komponenten möglich ist. Dabei ist allerdings noch nicht die Zusammenarbeit auf Anwendungsebene sichergestellt.
- **Interoperability** ist die Zusammenarbeit der Komponenten auf der Anwendungsebene. Dies wird durch Festlegung der Datentypen erreicht.
- **Interchangeability** gewährleistet die funktionale Gleichheit[20] aller Geräte, unabhängig des jeweiligen Herstellers. Erst bei diesem Grad der Offenheit kann ein Gerät ohne Änderungen am System, des Steuerungsprogramms etc. durch ein anderes ausgetauscht werden.

In der Verfahrenstechnik waren in der Vergangenheit proprietäre Prozessleitsysteme der Normalfall. Bei diesen konnten alle Komponenten von einem Anbieter mit ausreichend breitem Produktspektrum bezogen werden. Dabei begab sich der Kunde durch die Festlegung auf ein bestimmtes System in dauerhafte Abhängigkeit zum jeweiligen Systemlieferanten. Die Schnittstellen zur Erweiterung des Systems waren herstellerspezifisch, sodass eine nachträgliche Erweiterung der Anlage mit Komponenten anderer Hersteller ausgeschlossen war. Der Anbieter bestimmte den Preis ohne unmittelbaren Druck durch Mitbewerber.

Durch die Standardisierung der Kommunikation auf unterster Ebene wurde der Grundstein für den Weg hin zu offenen Systemarchitekturen gelegt. Die Entwicklung und Standardisierung von offenen und feldtauglichen Kommunikationseinrichtungen[21] sorgte für die allmähliche Ablösung der bis dahin proprietären Kommunikationstechnik. Bei diesen Feldbussystemen ist die Spezifikation für alle Herstellerfirmen frei zugänglich. Hierdurch ergeben sich unmittelbare Vorteile für den Anwender [BIRKHOFER 01]:

- Geräte verschiedener Hersteller können miteinander kommunizieren (Aufbau von so genannten Multivendoranlagen).

---

[20] Erreichbar durch so genannte Geräteprofile wie PA-Profil, Profidrive, CANOpen etc.
[21] so genannte Feldbusse, wie Profibus-DP, Interbus-S etc.

- Der Anwender muss sich nicht auf einen bestimmten Gerätehersteller bei der Realisierung seiner Anlage festlegen.
- Höherer Investitionsschutz, da das System von einem Verbund aus Herstellern unterstützt wird.
- Geräte verschiedener Hersteller werden bezüglich des Preises und der Funktionalität vergleichbar.

Bezüglich der Funktionalität ist allerdings noch sehr wenig standardisiert. Einen ersten Ansatz hierzu bieten die Profillösungen[22]. Allerdings sind die Parametergruppen nur bezüglich der Semantik beschrieben. Die Ausführung erfolgt über das Lesen und Schreiben von Parametern. Einen Schritt weiter geht das DeKOS-Bedienmodell (s. Kapitel 3.2), das real nutzbare Hilfsfunktionen dem Bediener zur Verfügung stellt.

2.2.2 Allgemeine Aufgaben des Engineerings

Der Begriff „Engineering[23]“ wird von verschiedensten Gruppen im Ingenieurwesen unterschiedlich verstanden und verwendet. Wie aus der Übersetzung des englischen Wortes zu erkennen ist, kann jede Tätigkeit an einer Produktionsanlage, von der Konzeption bis hin zur Diagnose, mit diesem Begriff belegt werden. Dies führt zwingenderweise zur Verwirrung, da noch dazu jeder Personenkreis den Begriff meistens mit seinen eigenen Vorstellungen verbindet. In Abbildung 2-4 ist der vollständige Lebenszyklus eines Produkts von der Produktidee bis zur Entsorgung allgemein nach ANDERL gegeben [ANDERL 02]. Ein vollständig geschlossener Lebenszyklus ist allerdings nur teilweise möglich.

[22] Funktionale Gleichheit der Hilfsfunktionalität in Form von Parametern für eine bestimmte Geräteklasse unabhängig des Herstellers durch Vereinheitlichung der Geräteparameter.

[23] deutsche Übersetzung: Ingenieurwesen, Konstruktion, Technik

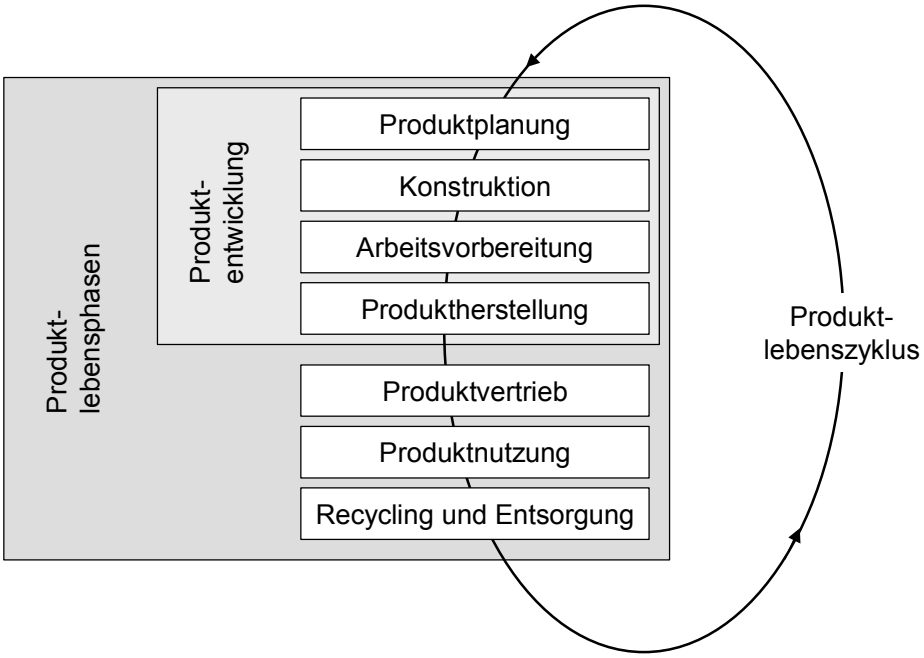


Abbildung 2-4: Produktlebenszyklus [ANDERL 02]

Die **Produktlebensphasen** bedeuten dabei [ANDERL 02]:

- Die **Produktplanung** ist eine unternehmerische Aufgabe und umfasst die Festlegung und Beschreibung neuer Felder der Produktentwicklung. Darin enthalten ist die Auslösung der Entwicklungs- und Konstruktionsaufträge.
- Die **Konstruktion** ist die erste technisch-planerische Phase. Diese umfasst sowohl Entwicklungstätigkeiten als auch experimentelle Tätigkeiten zur Informationsbeschaffung über physikalische Sachverhalte und die vollständige Beschreibung eines zukünftigen Produkts.
- Die **Arbeitsvorbereitung** umfasst die Arbeitsplanung und -steuerung. Die Arbeitsplanung stellt die zweite technisch-planerische Phase dar, deren Schwerpunkte in der Fertigungs-, Montage- und Prüfplanung liegen. Als Arbeitssteuerung (oft auch als Fertigungssteuerung) bezeichnet man die Gesamtheit der überwiegend organisatorischen Funktionen zur Steuerung des Ablaufs in der Fertigung.
- Die **Produktherstellung** wandelt die bisher nur als Pläne vorliegenden Informationen über technische Produkte mittels Materie und Energie in physikalisch existente Produkte. Hierbei wird der vorher formlosen Materie mittels Energie eine geplante Gestalt gegeben.

- Der **Produktvertrieb** ist die sich anschließende Brücke vom Hersteller zum Nutzer. Transport, Lagerung, Verteilung, Kosten und Termine sind Merkmale, die den Vertriebsaspekt kennzeichnen.
- Die **Produktnutzung** ist die eigentliche Zielphase für die das Produkt geschaffen wurde. Hierunter fällt nicht nur der vorgesehene Gebrauch des technischen Produkts, sondern ebenso die Wartung und Pflege sowie die Instandhaltung, Diagnose und eine eventuelle Reparatur.
- Die **Produktrecycling- und Entsorgungsphase** legt fest, was mit dem Produkt nach Ablauf der Nutzungsdauer geschieht. Vielfältige Prozesse führen zu Verschleiß oder zur technischen Veralterung. Verschrottung und Recycling sind grundsätzliche Möglichkeiten der Produktbeseitigung.

Ein erster Ansatz zur Ordnung der Begriffswelt im Bereich der Produktnutzung ist nach BENDER/KUTTIG/MEYER in [BENDER/KUTTIG/MEYER 02] gegeben. In die dort vorgenommene Analyse sind sowohl der Gerätelifecycle als auch die Hierarchieebenen einer komplexen Fertigungsanlage eingeflossen. Das daraus resultierende zweidimensionale Engineeringmodell (s. Abbildung 2-5) soll im Folgenden kurz erläutert werden.

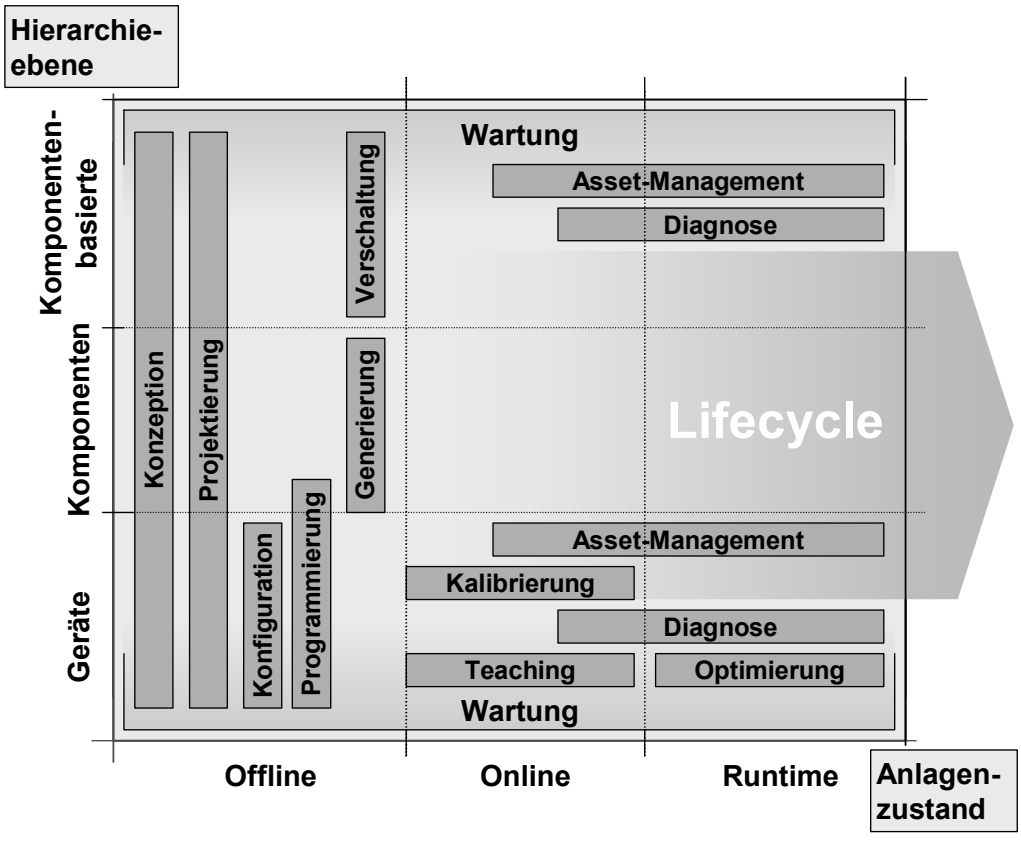


Abbildung 2-5: Zweidim. Engineeringmodell [BENDER/KUTTIG/MEYER 02]

Das Engineering, bezogen auf die einzelnen Lebensphasen, erfolgt entweder im Offline-, Online- oder Runtime-Modus der Anlage. Zum Offline-Engineering zählen beispielsweise die Projektierung, Programmierung, Konfiguration und die Grundparametrierung. Auch zählt hierzu das Verschalten von Partialsystemen zu einem Gesamtverbund. Dem Online-Engineering ist das Kalibrieren eines Sensors zuzuordnen, das Teaching (Positionieren eines Antriebes) etc. Im Runtime-Modus der Anlage erfolgt dann das Optimieren des Prozesses, Asset-Management Funktionen wie die Analyse von Prozessdaten, Trendverfolgung etc.

Die Hochwertachse des Modells wird repräsentiert durch die Hierarchieebenen einer Anlage. Auf der untersten Ebene befinden sich die einzelnen Feldgeräte. Aus diesen Geräten werden in der mittleren Schicht durch geeignete Engineering-Maßnahmen Komponenten gebildet. Diese gezielte Zusammenfügung der Feldgeräte geschieht in Anlehnung an die zu erfüllende Automatisierungsaufgabe und ist vom Konstrukteur vorgegeben. Auf der obersten Ebene erfolgt abschließend, im Sinne der Kommunikationsherstellung, die Verschaltung zwischen den einzelnen Komponenten.

2.2.3 Engineering von Partialsystemen

Ohne in diesem einführenden Kapitel näher auf den Lifecycle von Partialsystemen einzugehen und die daraus resultierenden Engineering-Aufgaben vollständig zu analysieren, soll der betrachtete Bereich eingegrenzt werden. Dabei spielt die Anwendergruppe[24] der zu betrachtenden Anlage eine wesentliche Rolle. Bei vielen Tätigkeiten des Engineerings ist das Interesse des Bedieners auf die Gesamtstruktur der Anlage gerichtet. Die einzelnen Feldgeräte spielen dabei nur eine untergeordnete Rolle [NE 91]. Diese erfüllen in der Kombination eine Gesamtaufgabe die für den Bediener, beispielsweise bei der Diagnose, relevant ist [BLUM 02].

Bei der Inbetriebnahme, Optimierung oder dem Produktwechsel einer Produktionsanlage wird in die Geräteparametrierung eingegriffen. Dabei ergeben sich häufig Abhängigkeiten zwischen den einzelnen Geräten, die dem Personal beim Engineering bekannt sein „sollten“. Diese sind aus der Konstruktion bekannt und werden in der Dokumentation festgehalten [RICHTER 03]. Ist dies nicht vollständig der Fall, wird in der Praxis nicht selten nach dem „Trial&Error“-Prinzip vorgegangen [TOMASIK 01]. So haben Änderungen an den Einstellungen eines Gerätes häufig Auswirkungen, die zu weiteren Änderungen führen.

[24] unterteilt in so genannte Benutzerrollen [VDI 2187]

Soll beispielsweise die Transportgeschwindigkeit eines Stetigförderers, bestehend aus *n* baugleichen dezentralen Antrieben (s. Abbildung 2-6), variiert werden, so muss hierzu der Service-Mitarbeiter mit dem dazu erforderlichen Bedientool alle Frequenzumrichter einzeln den neuen Begebenheiten anpassen. Eventuell ist er dabei in der Lage, einen Parametersatz zu erstellen, der dann allerdings noch auf die *n* Antriebe manuell zu übertragen ist. Verfügt das Bedientool nicht über die Möglichkeit Parametersätze abzuspeichern, ist sogar die Parametereingabe an allen *n* Frequenzumrichtern einzeln notwendig.

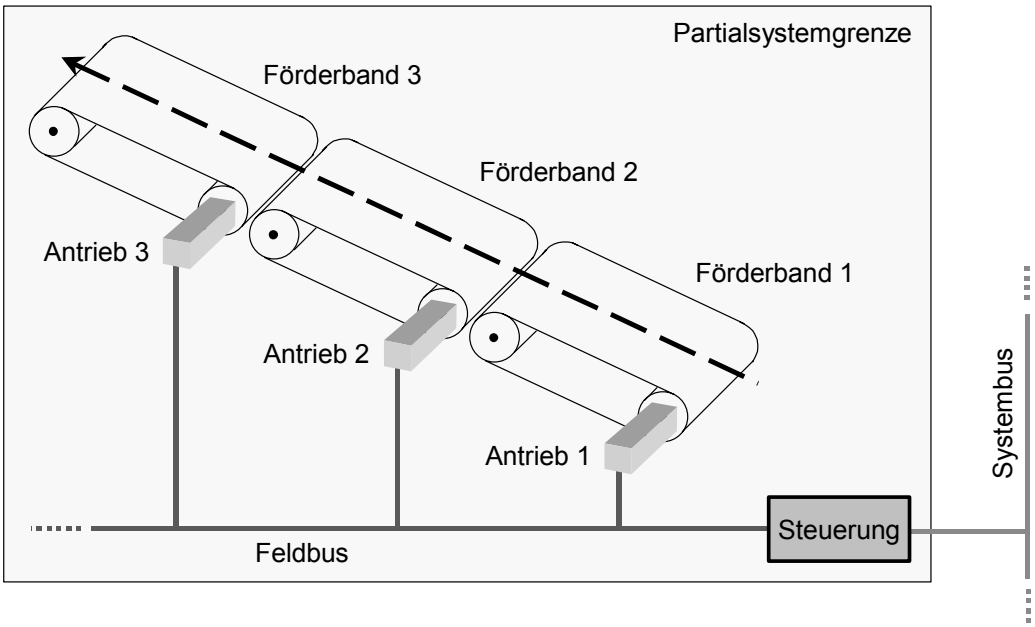


Abbildung 2-6: Schematischer Aufbau eines Stetigförderers

Es existieren diverse Ansätze die sich mit dem Engineering auf Partialsystemebene befassen. In der Verfahrenstechnik [MÜLLER ET AL. 02], [NICKLAUS/FUß 00], [KOSCHEL/MÜLLER/NICKLAUS 00] werden im Rahmen des Asset Managements[25] zusätzliche Diagnosemöglichkeiten und Instandhaltungsmaßnahmen von Feldgeräten geschaffen, um das Engineering von Partialsystemen zu vereinfachen. Dabei soll eine Effizienzsteigerung und Kostenminimierung bei der Fertigung u. a. durch schnelles Auffinden und Beseitigen von Störungen erreicht werden [NICKLAUS/FUß 00]. In der Fertigungstechnik sind solche Entwicklungen derzeit nicht zu erkennen.

[25] Erkennen und Erhalten des technischen Anlagenzustandes mit dem Ziel, die geforderte Verfügbarkeit bei minimalem Aufwand sicherzustellen [KOSCHEL/MÜLLER/NICKLAUS 00].

2.2.4 Beispiel der Inbetriebnahme eines Portals

Anhand des nachfolgenden einfachen Partialsystems sollen die notwendigen Engineering-Maßnahmen von der Inbetriebnahme bis zur Umprojektierung und die zu beachtenden Abhängigkeiten zwischen den Feldgeräten beispielhaft erläutert werden. In einer fertigungstechnischen Anlage zur Serienprüfung von Reflexionslichtschranken befindet sich ein Hochregallager, indem die zu prüfenden Komponenten eingelagert sind. Das Hochregallager verfügt über ein zweiachsiges Portal zum Transport von Paletten, auf denen sich die Prüflinge befinden (s. Abbildung 2-7).

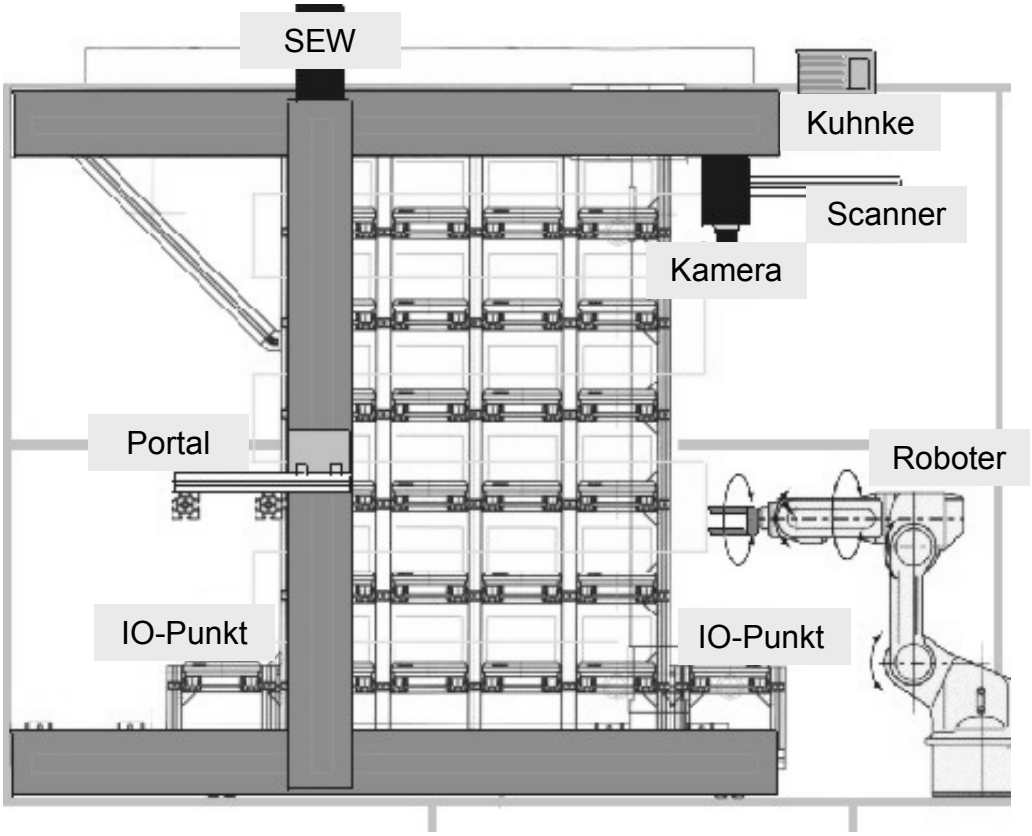


Abbildung 2-7: Schematische Darstellung des Hochregallagers

Der Fokus der nachfolgenden Betrachtungen beschränkt sich zur Vereinfachung auf das Partialsystem Portal mit zwei Antrieben und dem Schieber zum Auf- und Abladen (s. Abbildung 2-8). Bei der Konstruktion wurde für die Rechtswertachse ein Antrieb der Firma Kuhnke gewählt. Dieser entsprach den technischen Anforderungen und war vom Verhältnis zwischen Baugröße und Antriebsleistung optimal. Für die Hochwertachse wurde ein Antrieb der Firma SEW mit Lamellenbremse gewählt, da ein vergleichbarer Typ vom Hersteller Kuhnke nicht verfügbar war.

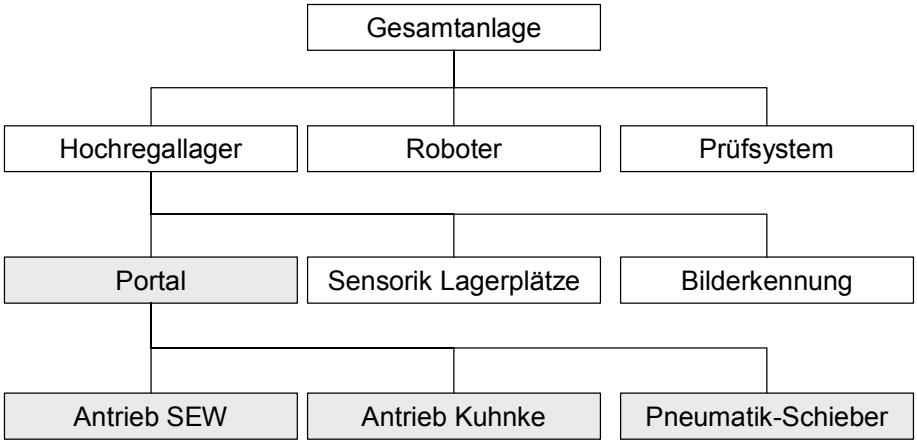


Abbildung 2-8: Vereinfachte Struktur der Prüfanlage

Die Anforderungen an die zwei Antriebe waren so unterschiedlich, dass es dem Konstrukteur mit angemessenem Aufwand nicht möglich war, zwei Antriebe eines Herstellers einzusetzen. Dementsprechend werden für das Engineering zwei unterschiedliche Tools benötigt (s. Abbildung 2-9, Abbildung 2-10). Die Hersteller bieten diese zusammen mit den Geräten an.

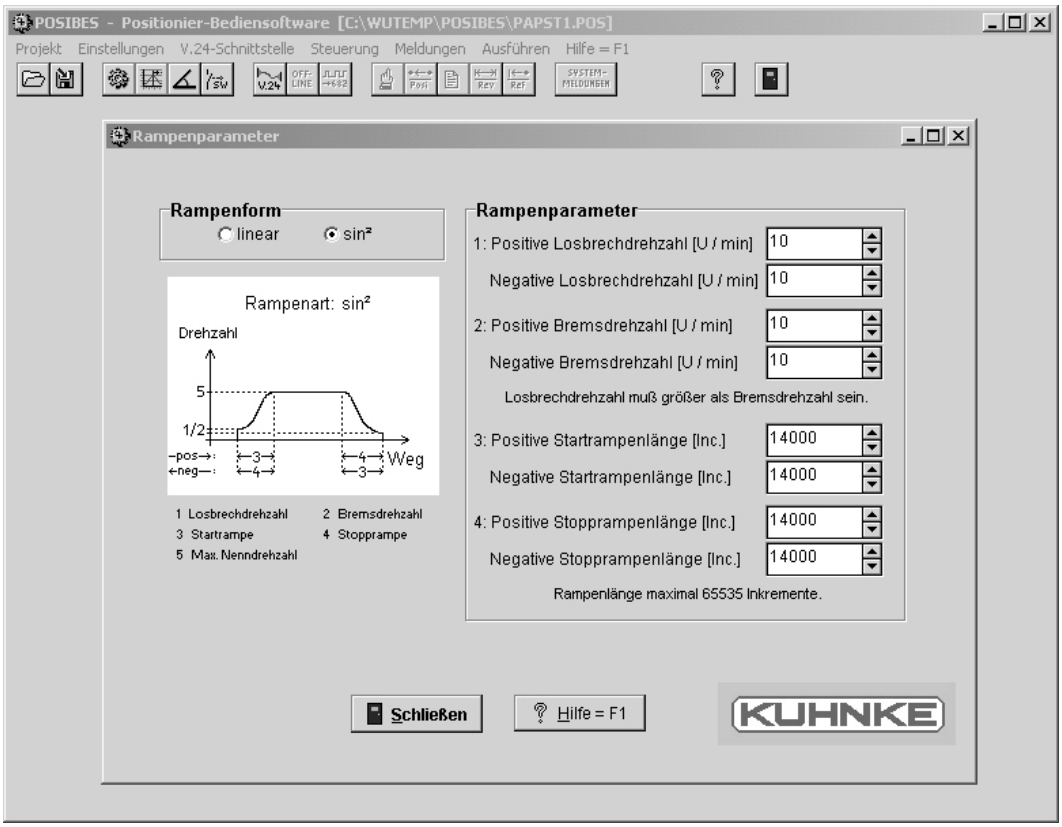


Abbildung 2-9: Bedientool der Firma Kuhnke

Die Oberflächen unterliegen den verschiedenen Bedienphilosophien der Hersteller. Eine Einarbeitung in beide Softwaresysteme ist daher unumgänglich. Eine Copy&Paste-Funktion, wie sie bei jeder Office-Anwendung existiert, ist nicht verfügbar. Erschwerend kommt hinzu, dass die Hilfsfunktionalität der zwei Geräte durch unterschiedliche Parameter auszuführen ist.

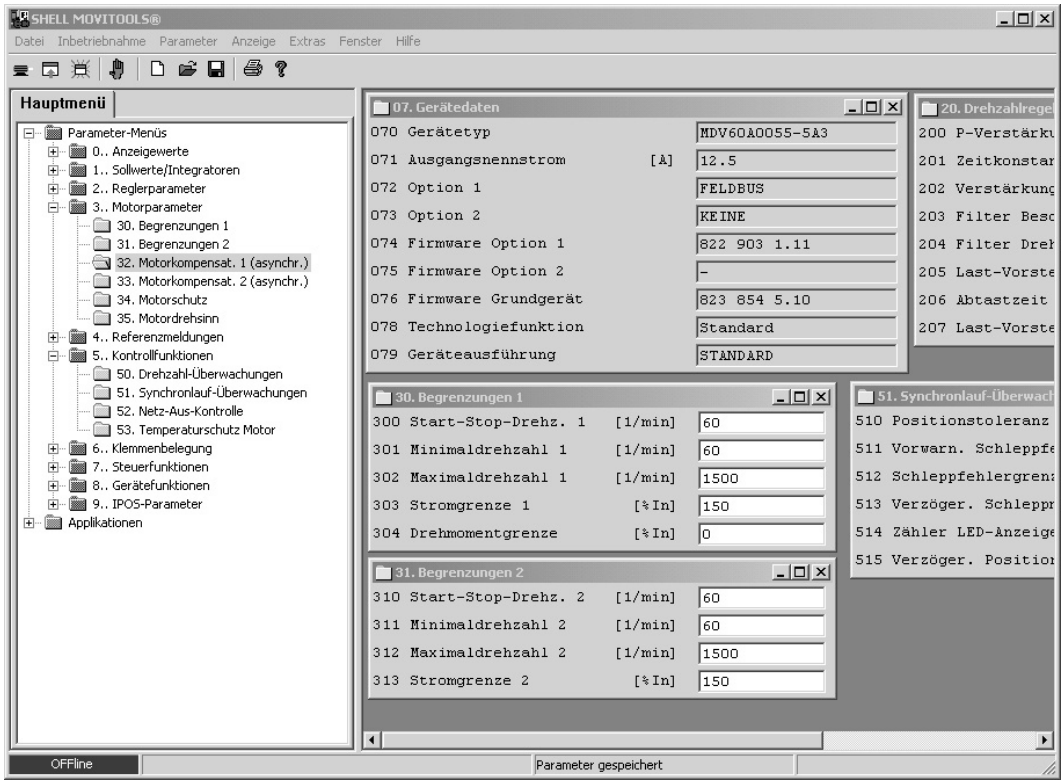


Abbildung 2-10: Bedientool der Firma SEW

Die Kommunikation zwischen Feldgerät und Bedientool erfolgt in beiden Fällen über die serielle Schnittstelle (RS-232) eines Servicelaptops[26], auf dem die Software installiert wurde. Der Schieber wurde als doppeltwirkender Pneumatik-Zylinder mit verstellbarer Drossel realisiert. Diese verfügt wahlweise über eine manuelle oder elektrische Verstellung. Hierdurch kann die Ein- und Ausfahrgeschwindigkeit des Zylinders stufenlos variiert werden. Ein eigenes Softwaretool wird nicht benötigt.

Dem Inbetriebnahmepersonal sind aus der Konstruktion folgende Daten bekannt:

- Max. zulässige Verfahrgeschwindigkeit: 0,30m/s
- Max. Beschleunigung (Anfahren und Bremsen): 0,45m/s^2

[26] Auch als Programmiergerät etc. bezeichnet.

- Sin^2-Rampe empfohlen
- Min. Positioniergenauigkeit: ±1,5mm
- Übersetzungsverhältnis der Linearantriebe: re. 1U=14mm, ho. 1U=10mm
- Detaillierte Motordaten (Phasenzahl, Nennspannung etc.)

Bei der Inbetriebnahme des Portals muss sich der Bediener zunächst in die unterschiedlichen Philosophien der zwei Tools einarbeiten. Durch die unterschiedlichen Architekturen ist wie zuvor erwähnt auch kein Export/Import der Rampendaten von einem Antrieb auf den anderen möglich. Bei der Inbetriebnahme ist folgendermaßen vorzugehen:

1. Verbinden des Servicelaptops mit dem Kuhnke-Antrieb und Start des Bedientools Posibes
2. Eingabe der Motordaten
3. Berechnung und Eingabe der max. Drehzahl unter Berücksichtigung des Übersetzungsverhältnisses
4. Berechnung und Eingabe der Rampendaten (Losbrechdrehzahl, Rampenlänge in Inkrementen und Rampentyp)
5. Eingabe der Positioniergenauigkeit (in Inkrementen)
6. Verbinden des Servicelaptops mit dem SEW-Antrieb und Start des Bedientools Movitools
7. Eingabe der Motordaten
8. Berechnung und Eingabe der max. Drehzahl unter Berücksichtigung des Übersetzungsverhältnisses
9. Berechnung und Eingabe der Rampendaten (Losbrechdrehzahl, Rampenlänge in Sekunden und Rampentyp)
10. Eingabe der Positioniergenauigkeit (in Inkrementen)
11. Einstellung der Fahrgeschwindigkeit des Pneumatik-Zylinders über die Einstellschraube der Drossel

Nachdem die Anlage in Betrieb gegangen ist, soll diese nach einigen Wochen auf einen neuen Prüflingstyp umgerüstet werden. Dieser ist höher und schwerer als der bisherige Typ, deshalb muss die maximale Beschleunigung reduziert werden, um ein Kippen und Herausfallen der gesteckten Prüflinge auf den Paletten zu verhindern. Zusätzlich wird die Anlage noch durch ein mechanisches Gitter abgesichert, sodass höhere Verfahrgeschwindigkeiten zulässig sind. Die geänderten Daten ergeben sich folgendermaßen:

- Max. zulässige Verfahrgeschwindigkeit: 0,50m/s
- Max. Beschleunigung (Anfahren und Bremsen): $0{,}30 m/s^2$

Durch die Umstellung der Prüfanlage muss u. a. das Portal umprojektiert werden. Hierzu ist durch das Personal entsprechend vorzugehen:

1. Verbinden des Servicelaptops mit dem Kuhnke-Antrieb und Start des Bedientools Posibes

2. Berechnung und Eingabe der max. Drehzahl des Kuhnke-Antriebs unter Berücksichtigung des Übersetzungsverhältnisses
3. Berechnung und Eingabe der Rampendaten (Losbrechdrehzahl, Rampenlänge in Inkrementen und Rampentyp)
4. Verbinden des Servicelaptops mit dem SEW-Antrieb und Start des Bedientools Movitools
5. Berechnung und Eingabe der max. Drehzahl des SEW-Antriebs unter Berücksichtigung des Übersetzungsverhältnisses
6. Berechnung und Eingabe der Rampendaten (Losbrechdrehzahl, Rampenlänge in Sekunden und Rampentyp)
7. Einstellung der Fahrgeschwindigkeit des Pneumatik-Zylinders über die Einstellschraube der Drossel

Nach dem Testbetrieb und einer eventuellen iterativen Wiederholung der Umprojektierung ist die Anlage anschließend erneut einsatzbereit.

Das Beispiel hat anschaulich gezeigt, dass die Kopplung der Geräte durch den Fertigungsprozess Abhängigkeiten schafft, die sowohl bei der Inbetriebnahme, als auch bei einer Umprojektierung zu berücksichtigen sind. Da sich die Kopplung nicht in den Tools zur Bedienung wieder findet, muss sie der Bediener zum einen wissen und zum anderen beachten. Erschwerend kommt hinzu, dass die physikalischen Größen wie z. B. die Rampenlänge bei einem Antrieb in Inkrementen und beim anderen Antrieb in Sekunden anzugeben ist.

2.3 Defizite bei der Bedienung komplexer Produktionsanlagen

2.3.1 Eigenschaften komplexer Produktionsanlagen

In den letzten beiden Jahrzehnten ist die Bedeutung von Software und Elektronik in Produktionsanlagen immens gestiegen. Dabei wurde zum einen durch den Preisverfall der IC-Bauteile der Grundstein für die einfache und kostengünstige Realisierung der benötigten automatisierungstechnischen Komponenten gelegt. Zum anderen hat der Wunsch der Kunden nach einem hohen Automatisierungsgrad der Produktionsanlagen die Entwicklungs- und Forschungsarbeiten der Hersteller der Geräte enorm beschleunigt.

Als Gründe für eine Steigerung des Automatisierungsgrades wären zu nennen [BENDER 02]:

- Qualitätssteigerung: Maschinen können in vielen Bereichen präziser arbeiten als der Mensch (Wiederholgenauigkeit).

- Humanisierung: Unangenehme oder monotone Arbeiten wie Mülltrennung, klassische Fließbandarbeit etc. werden durch ein Automatisierungssystem ausgeführt.
- Sicherheit: Zum Schutz des Menschen werden gesundheitsgefährdende und risikobehaftete Arbeiten durch die Maschine ersetzt. So wurden die ersten Roboter im Automobilbau bei der Rohkarosseriefertigung zum Punktschweißen eingesetzt.
- Rationalisierung: Eine Erhöhung des Automatisierungsgrades führt zur Steigerung der Produktivität.

Nach TOMASZUNAS weisen moderne komplexe Produktionsanlagen nachfolgende Eigenschaften auf [TOMASZUNAS 98]:

- Vereinigung vieler, hochgradig komplexer Funktionen unterschiedlicher Art (Handhabung, Transport, Speichern, Messen etc.) in einer Maschine
- Mehrere leistungsfähige Steuerungen
- Einsatz von intelligenten Feldgeräten und Feldbussystemen
- Klar erkennbare Maschinenstruktur hoher Komplexität mit mehreren Hierarchieebenen (Produktionsanlage, Partialsystem, Baugruppe, Komponente)

War in jüngster Vergangenheit besonders die Verfahrenstechnik noch gekennzeichnet durch eine Punkt-zu-Punkt Verdrahtung und eine zentrale Steuerung, etabliert sich in diesem Bereich zunehmend der Feldbus in Zusammenhang mit hoch komplexen aber flexiblen Feldgeräten [STUPP/OESTREICH 03]. Wie schon in Kapitel 2.2.1 erwähnt, gibt es diverse Bemühungen den Engineering-Aufwand dabei zu minimieren. Im Gegensatz hierzu wird die Komplexität der intelligenten Feldgeräte in der Fertigungstechnik immer noch stark unterschätzt. Zu diesen hoch komplexen Geräten zählen nicht nur die Antriebe, sondern auch beispielsweise Bilderkennungseinheiten, Wägesysteme u. ä., die meist mehrere Prozessfunktionen in einem Gerät anbieten und folglich auch durch eine Vielzahl von Hilfsfunktionen gekennzeichnet sind.

2.3.2 Anzahl und Abhängigkeiten der eingesetzten Bedientools

Zum Erhalt einer Anlage müssen unzählige Anwendungsprogramme installiert und gepflegt werden. Der Schulungs- und Einarbeitungsaufwand für das Personal ist immens gestiegen. Durch die Toolvielfalt wird die Anlagenbedienung unnötig verkompliziert, schnelle Reaktion bei eventuellen Störungen dem Personal erschwert. Eine zusätzliche Fehlerquelle durch eine Gerätebedienung mit dem falschen Tool, nicht Auffinden des passenden Parametersatzes bei Geräte-

austausch etc. ist gegeben. Dies führt zu einem erhöhten Kostenaufwand für den Betreiber, der sich mittelbar in den Herstellungskosten des Produkts niederschlägt [NICKLAUS/FUß 00]. Ansätze wie die FDT-Technologie versuchen dieses Problem zu lösen. Das Ziel ist dabei, nur noch eine Software, die so genannte Frameapplication (s. Definition 2.9) auf der Engineering-Workstation[27] verwalten zu müssen. Die einzelnen Feldgeräte werden dann über die vom Hersteller mit ausgelieferten Bedienproxys angesprochen (s. überlappende Fenster in Abbildung 2-11). Somit wird der Verwaltungsaufwand für die benötigte Software zwar minimiert, aber die Problematik der Bedienung der einzelnen Geräte wird nur von den proprietären Bedientools hin zur Proxy-Lösung verlagert. Oft wird die firmeneigene Bedienphilosophie bei der Realisierung des Bedienproxys vollständig übernommen [MEYER 01].

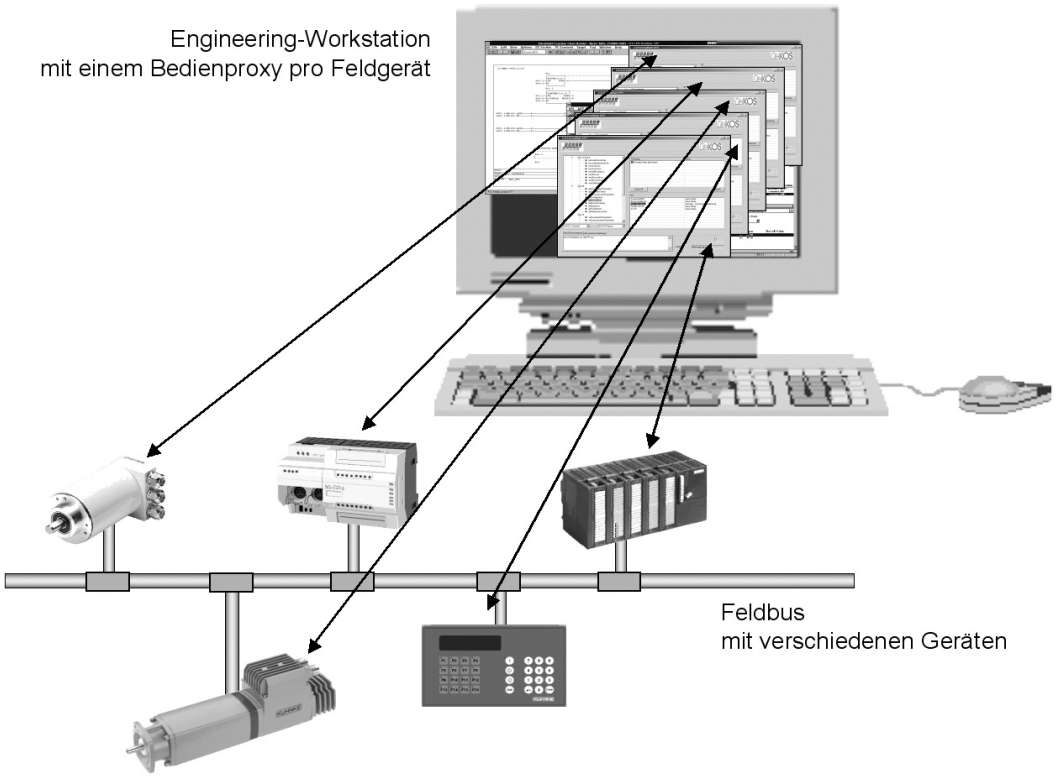


Abbildung 2-11: Komplexität bei der Gerätebedienung

Durch die Bemühungen, die Benutzerschnittstellen zu standardisieren, ist man noch einen Schritt weiter gegangen. Ansätze wie FDT-Styleguides zur Vereinheitlichung des Look&Feel oder die Standardisierung der Hilfsfunktionalität der

[27] Komponente (PC) zur Konfigurierung der Systemfunktionalität. Dient sowohl zur Programmierung der Steuerungen als auch zur Bedienung (im Sinne der Definition 2.7) der Feldgeräte [FRÜH 97].

Feldgeräte im DeKOS-Verbundprojekt haben zu einer nun möglichen, intuitiven Bedienung der Feldgeräte enorm beigetragen. Allerdings ist die hohe Anzahl der zu handhabenden, verschiedenen Bedienproxys bei einer komplexen Produktionsanlage nach wie vor geblieben (s. Abbildung 2-11).

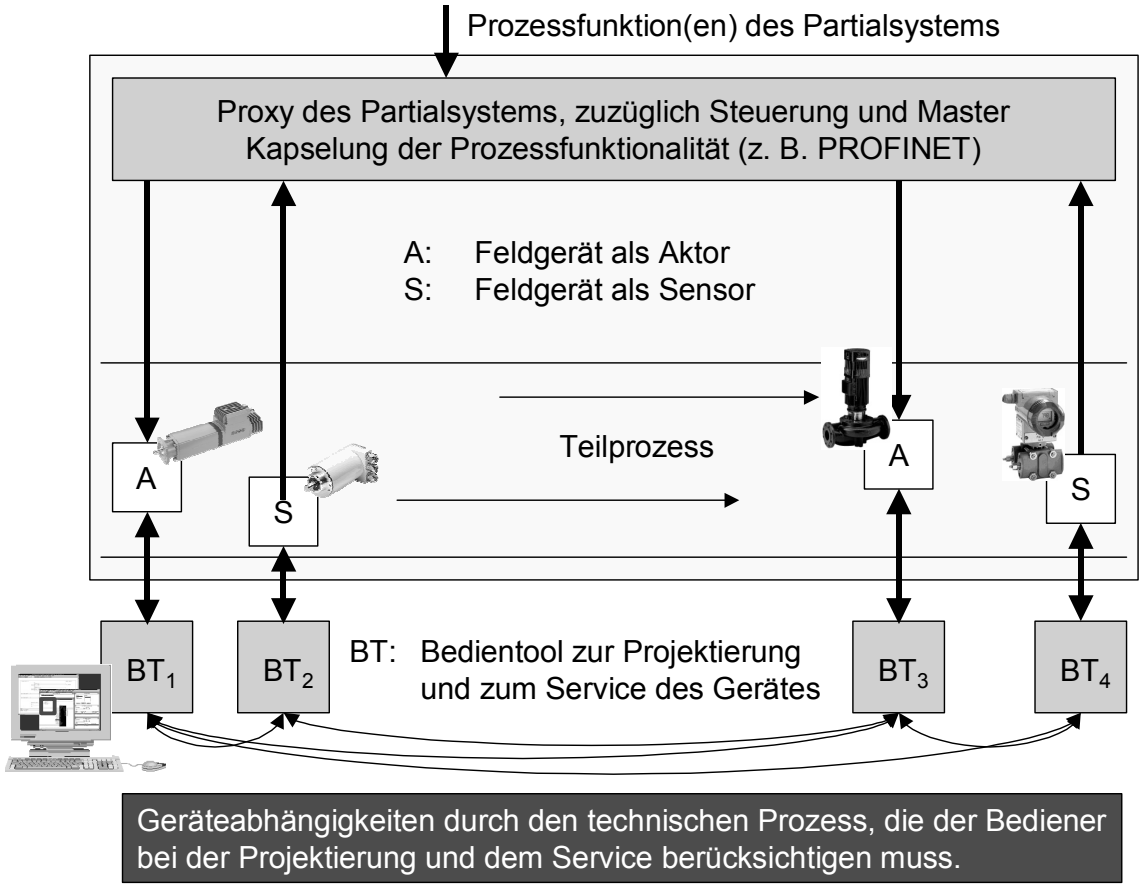


Abbildung 2-12: Geräteabhängigkeiten durch den technischen Prozess

Zur Kapselung der Prozessfunktionalität haben sich Technologien wie IDA, PROFInet etc. etabliert, die eine aufwandsarme Integration vorgefertigter Partialsysteme herstellerunabhängig ermöglichen. Die Realisierung der Prozessfunktionalität bleibt dabei dem Integrator verborgen. Zur Projektierung und zum Service des Partialsystems muss allerdings wieder auf herkömmliche Engineering-Methoden zurückgegriffen werden. Dabei wird jedes einzelne Feldgerät eigens betrachtet. Abhängigkeiten zwischen den einzelnen Geräten, die sich durch den technischen Prozess ergeben, finden nach wie vor durch die Bedientools keine Berücksichtigung (s. Abbildung 2-12) [MEYER 03]. Im Rahmen des Asset Managements existieren Ansätze, um durch eine gezielte Verknüpfung von meistens Prozessfunktionen zusätzliche Diagnosemöglichkeiten von Partial-

systemen zu schaffen. Hierzu werden zusätzliche Geräte in der Anlage benötigt, die wiederum Hilfsfunktionen aufweisen, die der Anwender handhaben muss und die Zusammenhänge darüber kennen muss. Als logische Konsequenz stellt sich für den Anwender die Frage, warum die Prozessfunktionen durch die Komponententechnologien gekapselt werden, jedoch die Hilfsfunktionen nicht.

2.3.3 Speziell notwendiges Know-how

Die Vielfalt der Tools und Geräte führt trotz Standardisierungsbemühungen dazu, dass eine Person allein eine Produktionsanlage nicht mehr in Stand halten kann. Bei einer Störung, einem Ausfall oder einer Umprojektierung aufgrund eines Produktwechsels sind meistens mehrere Spezialisten erforderlich. Diese verfügen über das notwendige Detailwissen aus der Konstruktion. Der Abstimmungsaufwand zwischen den beteiligten Personen ist erheblich.

Aus der Sicht der Endanwender kann nicht oft genug unterstrichen und betont werden, dass das Fehlen offener und tief greifender Standards im Bereich des Engineerings immer mehr Kosten verursacht [BENDER ET AL. 02]. Dies ist zur Zeit das größte Hindernis auf dem Wege, die Offenheit der Automatisierungssysteme und den Grad der Automatisierung auf ein höheres Niveau zu heben. Die Problematik wird vor dem Hintergrund folgender Fakten besonders deutlich sichtbar [ITM 02A]:

- Die Tendenz zur Dezentralisierung in der Automation wird immer größer. Immer mehr und immer intelligentere Geräte und Systeme wandern ins Feld, wodurch die Systemkomplexität steigt. Um die Komplexität zu reduzieren, kommen folglich Komponenten zum Einsatz, für die entsprechendes Know-how bereits vorhanden ist. Die Auswahl der in Frage kommenden Produkte von unterschiedlichen Herstellern wird hierdurch naturgemäß stark eingeschränkt.
- Der Endanwender muss mit sehr knappen Personalressourcen auskommen. Das Beschäftigen mehrerer hoch spezialisierter Fachleute wird unbezahlbar. Das Engineering muss deshalb schnell und effektiv durchgeführt werden, was wiederum zur Problematik des zuvor aufgeführten Punktes führt.
- Die schnell wechselnden Systemanforderungen (Hard – und Software) in der heutigen PC-Welt bringen zusätzliche Probleme bei der Handhabung proprietärer Systeme. Die umfangreiche Toolvielfalt, Variantenvielfalt und Vielfalt an Bedienphilosophien ist heutzutage in komplexen Anlagen kaum überschaubar.

Die Shell&DEA Oil GmbH als Projektpartner im Verbundprojekt DeKOS sieht die Situation wie folgt [ITM 02A]:

„Die Vielfalt der Geräte und Systeme führt dazu, dass gerade im Bereich des Engineerings dringend Standards benötigt werden. Die jetzige Situation führt dazu, dass der Endanwender „durch die Hintertür“ der proprietären Engineering-Werkzeuge und der proprietären Engineering-Funktionalität an den Lieferanten der Erstausrüstung einer Anlage gebunden wird. Dies ist das Ende von so genannten offenen Systemen, aller Standards in der Feldbuskommunikation zum Trotz. Bei Folgeaufträgen, Anlagenerweiterungen und Modernisierung fällt in der Regel die Kaufentscheidung zu Gunsten des Erstlieferanten: „Unsere Leute können ja damit umgehen“. Die eigentlichen wichtigen Fähigkeiten und Eigenschaften der Komponente, also die Prozessfunktionen, stehen nicht mehr im Vordergrund. Ein einheitliches Engineering bezogen auf die Hilfsfunktionalität schließt die Lücke um die Vorteile offener Systeme ausnützen zu können.“

Wie dem Beispiel aus Kapitel 2.2.4 zu entnehmen ist, liegt die Herausforderung bei der Erstinbetriebnahme und der Umprojektierung darin, genaue Kenntnisse über die Konstruktion wie beispielsweise die Übersetzungsverhältnisse zu haben. Diese notwendigen Informationen werden dem Personal bei der Erstinbetriebnahme höchstwahrscheinlich zur Verfügung stehen. Allerdings sind bei einem späteren Produktwechsel die notwendigen schriftlichen Unterlagen erfahrungsgemäß nicht mehr zur Hand.

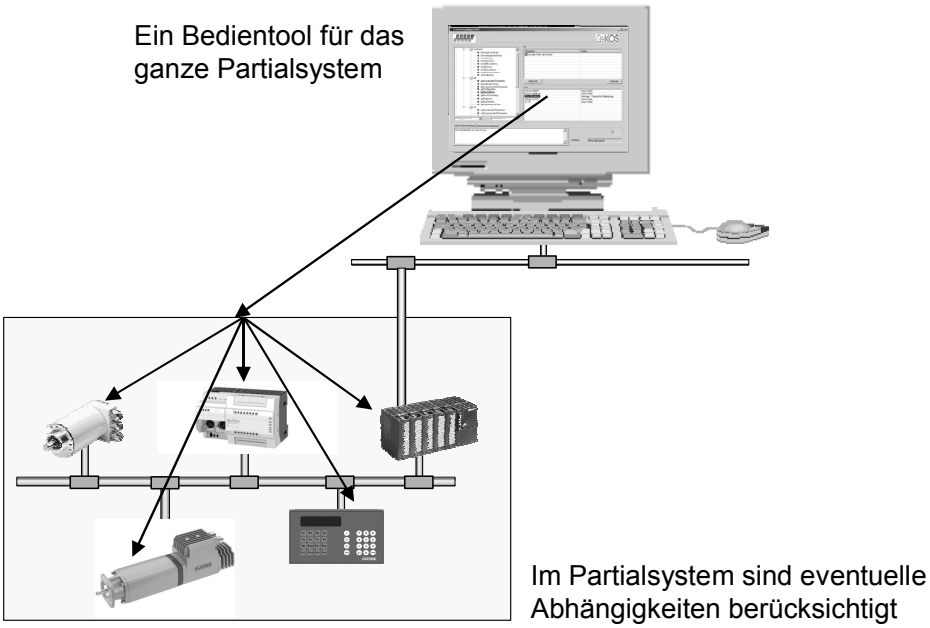


Abbildung 2-13: Kapselung der Hilfsfunktionalität von Partialsystemen

Eine nachträgliche Umprojektierung des Partialsystems könnte auch folgendermaßen ablaufen (s. Abbildung 2-13):

In einem Bedientool für das Partialsystem Portal (s. Kapitel 2.2.4) kann das Personal über eine zentrale Engineering-Workstation, die am Systembus der Anlage angeschlossen ist, direkt die Daten wie max. Geschwindigkeit, Beschleunigung eingeben. Das Bedientool kennt die konstruktiven Details und ändert automatisch die betreffenden Parametersätze der unterlagerten Geräte ab. Dabei ist dem Partialsystem bekannt, welche Geräte von der Änderung betroffen sind. Eventuelle Abhängigkeiten werden durch das Bedientool berücksichtigt. Somit werden nicht nur die Antriebe aus dem Beispiel den neuen Begebenheiten angepasst, sondern auch automatisch, ohne Wissen des Personals, die Einstellung der Drossel des Pneumatik-Zylinders geändert.

2.4 Zusammenfassung und Aufgabenstellung

Wie in Kapitel 2 dargelegt wurde, zeichnen sich moderne Produktionsanlagen sowohl in der Verfahrens- als auch in der Fertigungstechnik durch einen hohen Komplexitätsgrad aus. Dieser resultiert aus dem unüberschaubaren Funktionsumfang der intelligenten Feldgeräte. Die sich unmittelbar hieraus ergebenen Probleme betreffen primär den Endanwender der Geräte.

Dieser hat, wie in Kapitel 2.3 aufgezeigt, ein technisches Defizit zu kompensieren, indem er hohen Schulungsaufwand seines Personals in Kauf nimmt und bedingt durch die Unüberschaubarkeit der vielen Bedientools der Teilanlagen hohen Zeitaufwand bei Diagnose, Wartung und Umprojektierung akzeptiert. Hierdurch ergibt sich für den Anwender eine Bindung an bestimmte Hersteller, für dessen Geräte das notwendige Know-how vorhanden und der Engineering-Aufwand abschätzbar ist. Dies zieht ein ökonomisches Defizit nach sich, da die freie Wahl des Herstellers real nicht mehr existiert. Die Geräte, die man kennt, werden Geräten, die vom Preis-Leistungs-Verhältnis besser sind, aus besagten Gründen vorgezogen. Erhöhte Betriebskosten bei der Instandhaltung der Anlage sind die Folge, die sich auf die Herstellungskosten niederschlagen und letztendlich die Konkurrenzfähigkeit auf dem Markt beeinträchtigen. Eine wesentliche Voraussetzung für einen wirtschaftlichen Produktionsbetrieb ist auf Seiten des Endanwenders das Wissen über die zu erwartenden Betriebskosten. Vor Neuanschaffung von Anlagenteilen muss eine Prognose der zu erwarteten Gesamtkosten[28] des eingesetzten Automatisierungssystems über die gesamte Lebensdauer möglich sein. Wie DRESSLER und TRILLING in [DRESSLER/TRILLING 96] zeigen,

[28] Im betriebswirtschaftlichen Umfeld geprägt durch den Begriff Total Cost of Ownership.

übersteigen aber gerade die laufenden Kosten eines Systems die Investitionen der Anschaffung um ein Vielfaches und sind selten vorhersehbar.

Die Systemintegratoren, Anlagenbauer und Gerätehersteller sind gefordert, dem Wunsch der Anwender der Automatisierungskomponenten nach einer einfachen und standardisierten Bedienung der Partialsysteme Folge zu leisten. Denn erfolgreiche Innovationen orientieren sich am Markt [SCHULTZ-WILD/LUTZ 97], also an den Wünschen der Kunden, in dem Fall also der Endanwender automatisierungstechnischer Geräte. Im Widerspruch dazu kalkulieren aber gerade die Hersteller mit dem kostengünstigen Verkauf ihrer Geräte und den anschließenden Mehreinnahmen durch den OEM-Service[29] [DRESSLER/TRILLING 96]. Letztendlich ist eine Lösung nur denkbar, wenn der Wunsch des Kunden ohne großen Mehraufwand seitens der Hersteller realisierbar ist.

Vor dem Hintergrund dieser Problemstellung besteht die Aufgabe der vorliegenden Arbeit darin (s. Abbildung 2-14), eine Methode zur Kapselung der Hilfsfunktionalität von Partialsystemen zu entwickeln, sodass Abhängigkeiten zwischen den Bedientools der Geräte softwareseitig Berücksichtigung finden.

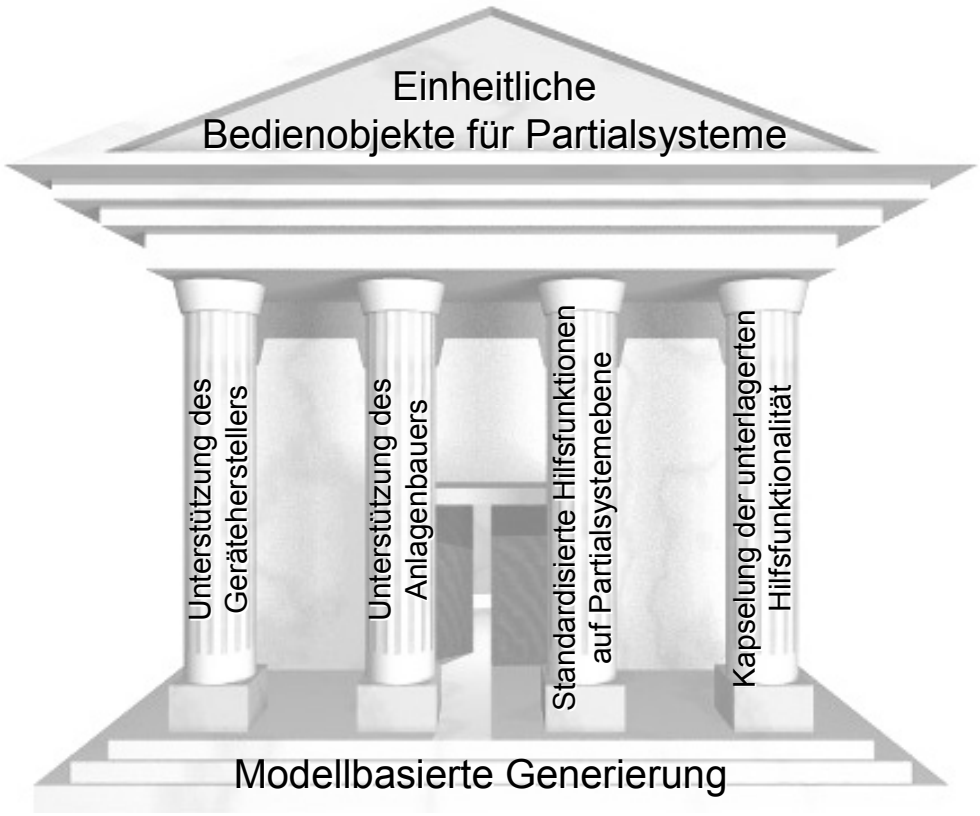


Abbildung 2-14: Vier Säulen der modellbasierten Generierung

[29] Original Equipment Manufacturers Service – Instandhaltung durch den Gerätehersteller

Hierdurch soll es ermöglicht werden, dass der Bediener Partialsysteme genauso wie einzelne Feldgeräte zentral und komfortabel, ohne Kenntnis der Interna über standardisierte Hilfsfunktionen bedienen kann. Dem Anlagenbauer wiederum soll durch das zu entwickelnde Verfahren ein Konzept an die Hand gegeben werden, das es ihm ermöglicht, ohne viel Mehraufwand dem Kunden ein Bedientool, das die aufgeführten Defizite kompensiert, anbieten zu können. Der Gerätehersteller ist durch das Modell ebenfalls zu unterstützen, da eine Lösung nur mit geeigneten Feldgeräten denkbar ist.

Das nachfolgende Kapitel beschäftigt sich mit der Suche nach existierenden Lösungen für das vorgestellte Problem. Dabei werden u. a. Richtlinien und Empfehlungen in Bezug auf das Engineering von Feldgeräten und Partialsystemen sowie Ansätze von komponentenbasierten Architekturen aus benachbarten Gebieten auf ihre Tauglichkeit hin untersucht. Aufbauend wird gezeigt, dass die in Kapitel 2 identifizierte Problemstellung mit den heute vorhandenen Methoden nicht lösbar ist. Abschließend wird der Lösungsansatz gegeben.

3 Engineering-Konzepte und Technologien

Bevor ein Lösungsansatz für das in Kapitel 2 identifizierte Problem bei der Bedienung komplexer Produktionsanlagen entwickelt wird, werden zuerst existierende Standards und Technologien, die zur Lösung des Problems herangezogen werden können, auf ihre Tauglichkeit hin untersucht. Es zeigt sich, dass die heute zur Verfügung stehenden Engineering-Methoden nicht ausreichen, um die zuvor beschriebenen technischen Defizite zu kompensieren.

Nach einer Analyse der Normen, Richtlinien und Empfehlungen zur Bedienung von Geräten und Systemen werden die Ergebnisse des DeKOS-Verbundprojekts erläutert. Des Weiteren wird auf technologische Trends aus dem industriellen Umfeld näher eingegangen.

Abschließend folgt eine Bewertung des analysierten Stands der Technik. Aufbauend wird der mögliche Ansatz zur Lösung des zuvor identifizierten Problems gegeben.

3.1 Normen, Richtlinien und Empfehlungen

Die nachfolgenden Ausführungen geben einen Überblick der gültigen Normen, Richtlinien und Empfehlungen wieder. Der Fokus ist dabei auf die Inbetriebnahme und Instandhaltung (Wartung, Diagnose) von Feldgeräten und Systemen gerichtet. Zusätzlich wird, aus dem Bereich der Software-Ergonomie, auf die Handhabung von Automatisierungskomponenten über Bedientools eingegangen.

3.1.1 DIN-Normen

DIN EN 13306 – Begriffe der Instandhaltung

Die Norm IEC 60050-191 (International Electrotechnical Vocabulary - Chapter 191: Dependability and quality of service) wurde vom DIN[30] als Grundlage für die Ausarbeitung der Norm EN 13306 verwendet. Der Zweck dieser europäischen Norm ist die Definition der Grundbegriffe für alle Instandhaltungsarten

[30] Deutsches Institut für Normung e. V.

und für das Instandhaltungsmanagement. Mit Ausnahme der Instandhaltung von Software ist sie allgemeingültig.

Der Inhalt gliedert sich dabei folgendermaßen:

- Grundbegriffe
- Begriffe zu Betrachtungseinheiten
- Eigenschaften von Einheiten
- Ausfälle und Ereignisse
- Fehler und andere Zustände
- Instandhaltungsarten und -strategien
- Instandhaltungstätigkeiten
- Zeitbezogene Begriffe
- Instandhaltungshilfsmittel und -werkzeuge
- Wirtschaftliche und technische Richtgrößen

Diese Norm bildet die Voraussetzung für die Neufassung der nachfolgenden Norm DIN 31051.

DIN 31051 – Grundlagen der Instandhaltung

Die Norm DIN 31051 legt Grundlagen der Instandhaltung fest. Sie gliedert die Instandhaltung vollständig in die Grundmaßnahmen Wartung, Inspektion, Instandhaltung und Schwachstellenbeseitigung. Ergänzend folgen Definitionen, die mit den Begriffen aus der Norm EN 13306 zum Verständnis der Zusammenhänge notwendig sind.

3.1.2 VDI/VDE-Richtlinien

Systematisch hat der VDI/VDE[31] ein technisches Regelwerk aufgebaut, das heute mit über 1600 gültigen Richtlinien das breite Feld der Technik weitgehend abdeckt. Entsprechend den technologischen Entwicklungen werden die bestehenden Richtlinien regelmäßig aktualisiert und es kommen ständig neue hinzu.

VDI/VDE-Richtlinien sind richtungweisende Arbeitsunterlagen für den praktischen Arbeitsalltag. Mit ihren Beurteilungs- und Bewertungskriterien geben sie fundierte Entscheidungshilfen und bilden einen Maßstab für einwandfreies technisches Vorgehen.

[31] Verein Deutscher Ingenieure e. V., Verband Deutscher Elektrotechniker e. V.

VDI/VDE 2186 – Einheitliche Anzeige- und Bedienoberfläche für Antriebsregelgeräte

Die Richtlinie VDI 2186 wurde von einem Fachausschuss der GMA[32] erarbeitet, um einheitliche Benutzeroberflächen für Antriebe in der Prozess- und Fertigungstechnik aufbauend auf der Richtlinie VDI 2187 zu gewährleisten. Der Fokus liegt dabei auf den Status-, Diagnose- und Parameterinformationen digitaler Antriebsgeräte. Hierdurch wird eine erhebliche Arbeitserleichterung bei der Inbetriebnahme und Wartung dieser Feldgeräte erzielt.

Ergänzt wird die Richtlinie durch Hinweise für den Softwareentwickler bei der Realisierung des Bedientools. Hierzu sind auch in Anlehnung an Styleguides[33] beispielhafte Screenshots[34] enthalten.

VDI/VDE 2187 – Einheitliche Anzeige- und Bedienoberfläche für digitale Feldgeräte

Die Richtlinie VDI 2187 enthält Richtlinien für die Anzeige-Bedien-Komponente digitaler Feldgeräte, die jedoch auch für jedes parametrierbare und kommunikationsfähige Sensor- und Aktorsystem gültig sind. Daher wird diese Richtlinie auch für Partialsysteme als gültig betrachtet. Der erste Teil der Richtlinie beschäftigt sich mit der Engineering-Umgebung, der zweite Teil mit der Anzeige- und Bedienoberfläche (ABO).

Für die Engineering-Umgebung wird eine so genannte Projektsicht gefordert, die das Erstellen, Umbenennen, Löschen und Kopieren von Projekten ermöglicht. Hier soll das komplette Projekt aus den einzelnen Feldgeräten zusammengefügt und die im zweiten Teil behandelte ABO gestartet werden. Ein weiterer Punkt im ersten Teil ist das Rollenmodell, in dem definiert wird, welcher Personenkreis bestimmte Hilfsfunktionen ausführen darf. Insgesamt werden vier Rollen (siehe Tabelle 3-1) unterschieden und diesen unterschiedliche Zugriffsberechtigungen zugeordnet.

---

[32] VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik
[33] z. B. DTM-Styleguide zur einheitlichen Gestaltung der Benutzeroberfläche bei FDT
[34] Bildschirmausdruck

Rolle	betriebsrelevante Parameter	Instandhaltungs-relevante Parameter	alle Parameter
Anlagenfahrer	lesen und schreiben	lesen	lesen
Betriebliche Instandhaltung	lesen und schreiben	lesen und schreiben	lesen
Gerätespezialist	lesen und schreiben	lesen und schreiben	lesen und schreiben
Anlagenplaner	lesen und schreiben	lesen und schreiben	lesen

Tabelle 3-1: Rollenmodell nach [VDI 2187]

In Tabelle 3-2 sind die einzelnen Punkte des zweiten Teils über die Gestaltung des ABO zusammengefasst.

Identifikation des ABO	Informationsfenster mit Produktname, Version, Hersteller usw.
Zugriffsberechtigung	Unterstützung des Rollenmodells
Offline-Parametrierung	Möglichkeit, das Gerät offline einzustellen
Plausibilisierungen	Check der Datentype, des Wertebereichs...
Datenkonsistenz	Überprüfung der Daten auf Konsistenz
Kennzeichnung von Parametern	Kennzeichnung jedes Parameters, ob er änderbar, geändert oder unverändert ist
Kennzeichnung der Datenquelle	Sind die aktuellen Daten aus dem Gerät oder von einem Datenträger
Vergleich der Daten	Vergleich der Datensätze auf Geräten und Datenträgern
Eingabegeräte	Mehrere Alternativen zur Bedienung (Maus, Tastatur, Touchscreen usw.)
Bedienstrukturen	Festlegung der Menüstruktur (Datei, Bearbeiten, Ansicht, Gerät usw.)
Warnhinweise	Erscheinungsbild von Warnungen und Hinweisen
Fehlerbehandlung	Fehlerhafte Eingaben dürfen nicht ins Gerät gelangen
Schreibmodi ins Gerät	Schreiben des kompletten Datensatzes, Schreiben nur geänderter Daten sofortiges Schreiben in das Gerät

Tabelle 3-2: Gestaltung der Anzeige- und Bedienoberfläche nach [VDI 2187]

Zur Gestaltung der ABO wird auf bestehende Styleguides aus der Windows-Welt verwiesen.

VDI/VDE 3850 – Nutzergerechte Gestaltung von Bediensystemen für Maschinen

Die Richtlinie VDI 3850 bezieht sich auf Software- und nicht auf Hardware-ergonomische Gesichtspunkte. Sie ist bei der Gestaltung von Bediensystemen für fertigungstechnische Maschinen als Ergänzung zur VDI 2187 zu sehen. Zweck dieser Richtlinie ist die Angabe von Regeln und Empfehlungen zur einheitlichen Gestaltung von Bediensystemen für Maschinen.

Diese Bediensysteme gestatten einen flexiblen Eingriff seitens des Benutzers aufgrund sich ändernder Aufgabenstellung sowie die geeignete Aufbereitung und Darstellung von Maschinen- und Prozessdaten. Ein Bediensystem nach VDI 3850 umfasst alle diejenigen Komponenten einer Maschine, die der Interaktion mit Benutzern dienen und von diesen wahrnehmbar sind.

Der Inhalt gliedert sich dabei folgendermaßen:

- Begriffsdefinition
- Vorgehen bei der Gestaltung eines Bediensystems
- Grundlagen zur Gestaltung von Bediensystemen
- Informationscodierung
- Bildzeichen
- Abkürzungen
- Informationen und Informationsklassen
- Dialoggestaltung

Die Richtlinie befasst sich schwerpunktmäßig mit Bediensystemen für Werkzeugmaschinen, Roboter etc. und ist nur teilweise gültig für die Hilfsfunktionalität im Sinne der Definition 2.7.

3.1.3 NAMUR-Empfehlungen

Die NAMUR[35] ist ein Verband von Anwenderfirmen aus dem Bereich der Prozessleittechnik. Hersteller von Prozessleitsystemen, Hardware oder Software können nicht Mitglied werden. Bei den NAMUR-Empfehlungen (NE) handelt es sich um Erfahrungsberichte und Arbeitsunterlagen, die die NAMUR für ihre Mitglieder aus dem Kreis der Anwender zur fakultativen Benutzung erarbeitet hat. Diese Papiere sind nicht als Normen oder Richtlinien sondern ergänzend zu diesen anzusehen.

[35] Normenarbeitsgemeinschaft für Mess- und Regeltechnik in der chemischen Industrie

NE 57 – Anzeige und Bedienoberfläche für statische Frequenzumrichter

Die Empfehlung NE 57 berücksichtigt die Anforderungen zur Anzeige- und Bedienoberfläche von statischen Frequenzumrichtern. Das Ziel ist die Verbesserung des Services bei der Inbetriebnahme und der Instandhaltung. Der Kern ist dabei eine hierarchische Strukturierung der notwendigen Hilfsfunktionen, die dem Anlagenpersonal zugänglich sein müssen.

Das Tätigkeitsfeld erstreckt sich von dem Anzeigen von Messwerten, Meldungen über die Parametrierung bis hin zu Servicefunktionalität. Jede Gruppe beinhaltet anschauliche Beispielfunktionen.

NE 91 – Anforderungen an Systeme für anlagennahes Asset Management

Bei verfahrenstechnischen Anlagen werden unter Assets die gerätetechnischen Komponenten der Anlage verstanden. Dabei wird beim Asset Management zwischen der Betriebs-/Prozessführung und dem anlagennahem Geräte-Management unterschieden. Hierzu zählen unter anderem Maßnahmen zur Diagnose, Wartung und Instandhaltung der Komponenten.

Die Empfehlung wendet sich sowohl an die Anwender und Hersteller von Asset Management-Systemen als auch an die Feldgerätehersteller. Es werden Hilfestellungen zur Abschätzung des Nutzens entsprechender Systeme gegeben. Der erforderliche Funktionsumfang wird dabei beschrieben um entsprechende Systeme spezifizieren zu können. Wünschenswertes aus Anwendersicht wird aufgeführt, um dem Auftraggeber die Möglichkeit zur Kontrolle bei der Realisierung zu geben.

In Bezug auf die Parametrierung der Feldgeräte wird die Möglichkeit zur Parametrierung von einzelnen Geräten sowie ganzen Teilanlagen gefordert. Hierzu sind die Parameterinformationen der Feldgeräte in einer Datenbank des Management-Systems zu archivieren. Da es sich bei NE 91 um eine relativ neue Empfehlung handelt, sind einige Forderungen enthalten, die mit derzeitigen technologischen Möglichkeiten noch nicht vollständig zu erfüllen sind.

3.2 DeKOS-Bedienmodell

Durch die Entwicklung leistungsfähiger, kleiner und preiswerter Bauteile im Mikroelektronikbereich wurde es möglich, Funktionalität aus der Leitebene in die Feldgeräte zu verlagern. Die so entstandenen intelligenten Feldgeräte besitzen Vorteile hinsichtlich ihrer Einsatzmöglichkeit und ihrer Flexibilität. Im

Rahmen des vom BMBF[36] geförderten Verbundprojekts DeKOS (Dezentrale Kooperierende Offene Mikrosysteme) wurde, in Zusammenarbeit von Forschungsinstituten und Industrie, ein neues Konzept zur Gerätebedienung geschaffen.

Durch die enorme Zunahme an Funktionalität werden Systeme aus intelligenten Feldgeräten im Gegenzug immer komplexer und sind für den Anwender nur noch schwer beherrschbar. Zwar liefern Gerätehersteller zur Bedienung, Parametrierung, Diagnose und Wartung Softwaretools, größtenteils sogar sehr gute, diese sind jedoch für den Anwender erst nach dem Studium der jeweiligen Bedienphilosophie einsatzfähig. Durch diese unterschiedlichen Bedienkonzepte und durch uneinheitliche Namensgebung von Hilfsfunktionen und Parametern werden die Vorteile der intelligenten Geräte teuer erkauft. Hier setzt DeKOS an, und bietet ein einheitliches Bedienmodell mit einem definierten Namensraum. Folgende Anforderungen wurden im Projekt definiert:

- Gleiche Bezeichnung für gleiche Hilfsfunktionen
- Präzise semantische Definition der Hilfsfunktionen
- Komfortable Hilfsfunktionen (vgl. Assistenten) für unerfahrene Bediener
- Rollenorientierte Sichten, die nur die benötigten Hilfsfunktionen zu Verfügung stellen
- Plattform- und kommunikationsunabhängige Gerätebeschreibung

Mit einem Bedienmodell, das diesen Anforderungen entspricht, können die erhöhten Engineeringkosten, die durch die intelligenten Feldgeräte entstanden sind, wieder gesenkt werden.

3.2.1 Paradigmenwechsel

Einer der Kernpunkte bei DeKOS ist der Paradigmenwechsel weg von der datenorientierten hin zur funktionsorientierten Sicht auf das Gerät. Dies ist die ursprüngliche Sicht des Bedieners. Er will nicht die einzelnen Bits und Bytes innerhalb des Gerätespeichers sehen, sondern Hilfsfunktionen des Gerätes angeboten bekommen. Wünscht dieser beispielsweise bei einem Profibus-PA Gerät eine Fehlermeldung auszulesen, muss er Slot 12 Index 2 aus dem Register des Gerätes lesen und das Ergebnis anhand von Tabellen dekodieren, anstatt eine einfach zu verstehende Funktion wie *getDeviceError* auszuführen.

Die Analyse der Gerätefunktionalität von verschiedensten Geräteklassen und Herstellern ergab eine Aufteilung in Prozessfunktionalität, d. h. die eigentliche

[36] Bundesministerium für Bildung und Forschung

Funktion, die das Gerät im Betrieb zu erfüllen hat, und Hilfsfunktionalität, d. h. diejenige Funktionalität die dazu dient, das Gerät in den Zustand zu bringen, in dem die Prozessfunktionalität optimal genutzt wird (siehe Definition 2.6, Definition 2.7). Dabei übersteigt die Menge der Hilfsfunktionen die Menge der Prozessfunktionen - meist nur eine pro Gerät - bei weitem. Hier erfolgte im Rahmen des Projekts eine Standardisierung sowohl über Geräteklassen als auch über Herstellergrenzen hinweg. Spezielles Know-how eines Herstellers liegt in der Regel ohnehin in der Prozessfunktionalität verborgen, z. B. wie schnell und präzise steht ein Messwert zur Verfügung, während die Funktionalität zur Bedienung eines Gerätes von jedem Hersteller gleichermaßen zur Verfügung gestellt werden muss.

3.2.2 Funktionsbibliothek

Die Analyse eines repräsentativen Gerätespektrums aus der Verfahrenstechnik und der Fertigungstechnik ergab über 700 Gerätefunktionen [BENDER ET AL. 02]. Aus diesen Funktionen wurden Gruppen mit semantisch gleichen Funktionen gebildet und für diese jeweils *eine* Hilfsfunktion mit einem eindeutigen Funktionsnamen definiert (s. Abbildung 3-1).

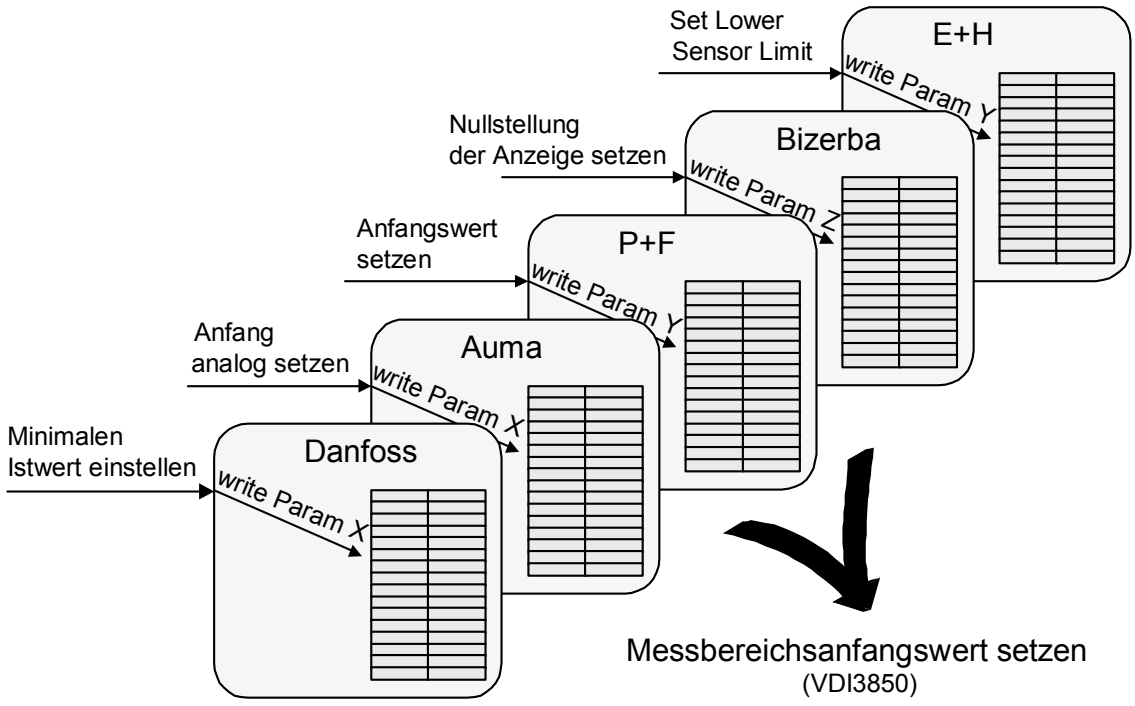


Abbildung 3-1: Semantisch gleiche Funktionen [BENDER ET AL. 02]

Die Definition besteht aus einem eindeutigen Namensraum und einer Beschreibung der Semantik. Damit die Funktionen bestimmten Anwenderkreisen und in verschiedenen Sprachen angeboten werden können, wurde zu jeder Funktion zusätzlich zu dem eindeutigen Namen ein ergonomischer Name, im weiteren als ErgoName bezeichnet, definiert. Die ErgoNames können in anderen Ländern durch die jeweiligen Sprachen sowie in bestimmten Domänen wie z. B. Papierindustrie oder Ölraffinerie etc. durch die dort gängigen Spezialbegriffe ersetzt werden. Als Ergebnis dieses Schritts wurden ca. 160 Hilfsfunktionen definiert, die als atomare Funktionen bezeichnet werden.

Neben den atomaren Funktionen werden noch Komfortfunktionen definiert. Im Gegenteil zu den atomaren Hilfsfunktionen können die Komfortfunktionen nur zum Teil aus der Gerätedokumentation entnommen werden, da sie meistens vom Anwender selber in Form von Notizen oder Makros entstehen. Die am Projekt beteiligten Firmen wurden mittels eines Fragebogens nach den Tätigkeiten, benutzten Tools und Funktionalität in verschiedenen Phasen des Lifecycles befragt. Im weiteren Schritt wurden die auf diesem Weg ermittelten Abläufe als Komfortfunktionen definiert.

Für die Komfortfunktionen werden, ähnlich wie bei den atomaren Hilfsfunktionen, die Bezeichner, die ergonomischen Namen und deren Bedeutung festgelegt. Die Abläufe werden jedoch nicht definiert, da sie sich in den einzelnen Geräten stark voneinander unterscheiden können. Beispielsweise wird der Ablauf einer Komfortfunktion *Kalibrierung durchführen* in einer Industriewaage anders definiert als in einem Radarfüllstandsmessgerät. Dies ist allerdings für den Bediener nicht von Bedeutung, da dieser einen bestimmten Vorgang auslösen will, die Details sind dabei nicht von Interesse.

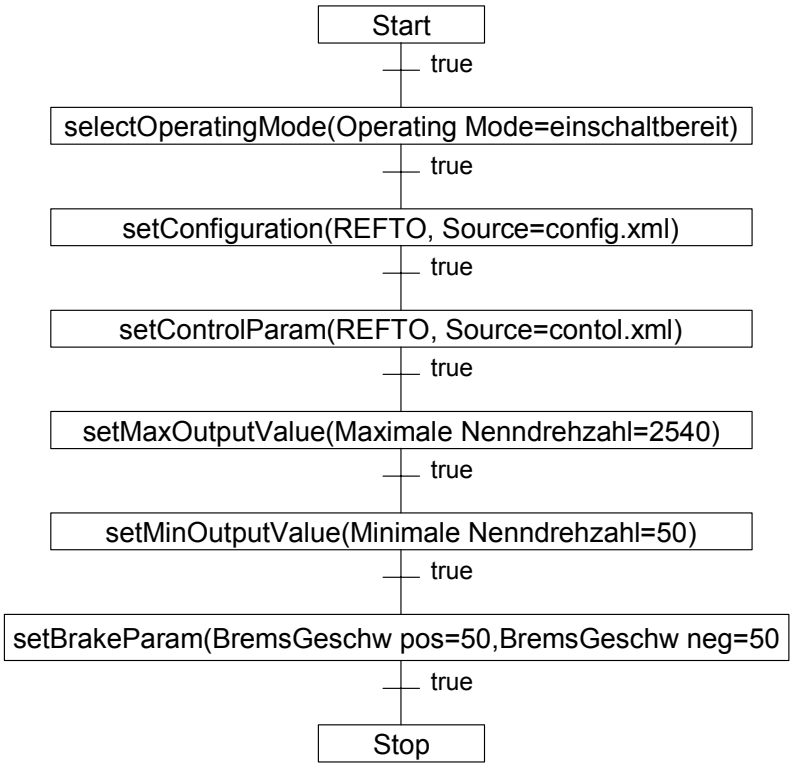


Abbildung 3-2: Komfortfunktion prepareOperation [BENDER ET AL. 02]

Das Beispiel in der Abbildung 3-2 zeigt die Definition der Komfortfunktion *prepareOperation* (ErgoName = Gerätebetrieb einstellen) für die Positioniersteuerung Kuhnke KUAX 682DP. Diese Komfortfunktion stellt eine Sequenz von Aufrufen einzelner atomarer Hilfsfunktionen dar, die eine komplette Parametrierung der Steuerung durchführen. Die Funktion *prepareOperation* kann beispielsweise nach einem Austausch der Steuerung aufgerufen werden, um eine neue Steuerung komplett zu parametrieren. Die Option *REFTO* vor dem Parameter Source bedeutet, dass der Übergabewert nicht direkt übergeben wird, sondern im angegebenen Pfad zu finden ist.

3.2.3 DeKOS-Spezifikation

Um den Anforderungen nach einem Rollenmodell gerecht zu werden, wird wie in der VDI/VDE 2187 vorgegeben eine Zuordnung von atomaren und Komfortfunktionen zu den einzelnen Bedienergruppen vorgenommen.

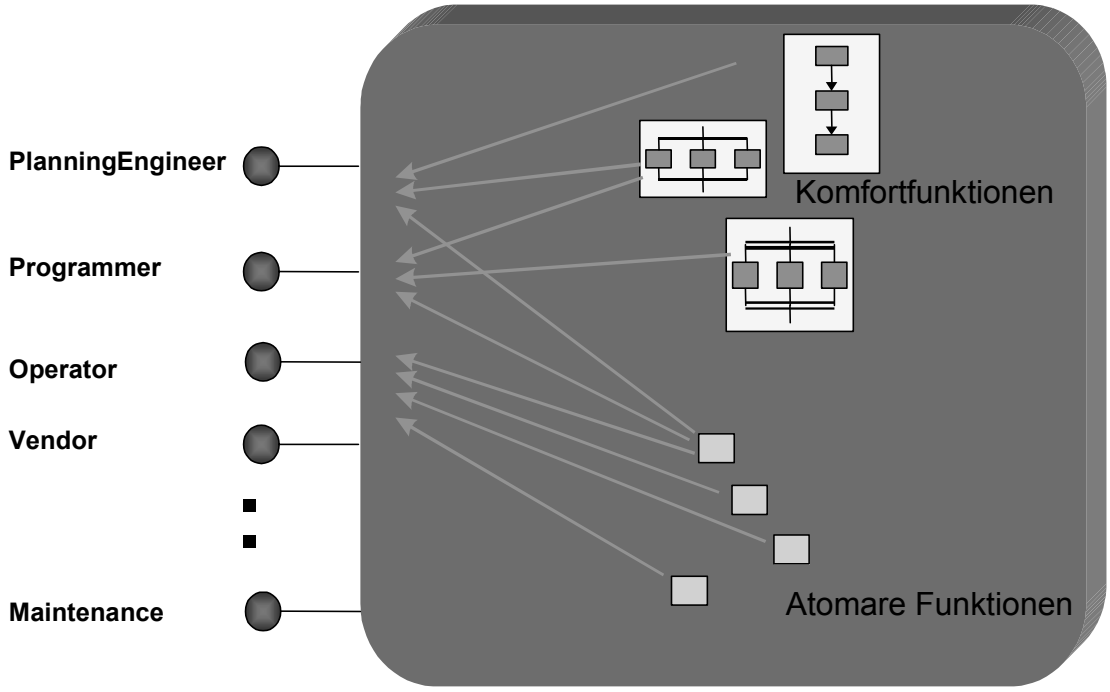


Abbildung 3-3: DeKOS-Bedienmodell [BREGULLA 01]

Abbildung 3-3 stellt das Zugriffsmodell dar. Die atomaren Hilfsfunktionen und die Komfortfunktionen werden über bestimmte Interfaces den entsprechenden Bedienerrollen zur Verfügung gestellt. Eine Person, die zu einem bestimmten Zeitpunkt eine bestimmte Rolle annimmt (z. B. durch das Anmelden am Bedientool), kann nur die für diese Rolle vorbestimmten Funktionen nutzen. Das Modell sieht ebenfalls das Erstellen eigener Rollen vor.

Die DeKOS-Spezifikation [ITM02] definiert das DeKOS-Bedienmodell mittels XML, wobei XML nicht zwingend vorgeschrieben ist, sich jedoch zur Beschreibung als geeignet erwies. In der Spezifikation wird definiert, wie eine DeKOS-konforme Beschreibung aufzubauen ist (s. Abbildung 3-4).

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <adml:DeKOSDEVICE DEVICENAME="Drive Control 628DP" SERNUM="0000" VENDOR="Kuhnke"
    xmlns:adml="http://www.itm.tum.de/2001/adml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.itm.tum.de/2001/adml DRA_V1.20.xsd">
    <FILEINFO AUTHOR="AZ; MBl" DATE="2002-07-25" FILETYPE="DDD" VERSION="1.20000">Allgemeines zur
      Datei</FILEINFO>
  - <PRODFUNCTION DEKOSNR="PF_1" NAME="move" FUNCTYPE="COMMAND">
    - <DeKOSDESC LANGUAGE="dt">
        <SPECDESC />
        <ERGONAME>Fahrauftrag</ERGONAME>
        <USERDESC>Fahrauftrag abschicken</USERDESC>
      </DeKOSDESC>
    </PRODFUNCTION>
  - <F2FUNCTION DEKOSNR="F2_2" FUNCTYPE="GET" NAME="backupDeviceData">
    - <DeKOSDESC LANGUAGE="dt">
        <SPECDESC>Gerätedaten, -Parameter oder gesamte Speicherblöcke, werden auf Datenträger
          gesichert.</SPECDESC>
        <ERGONAME>Gerätedaten sichern</ERGONAME>
        <USERDESC>Sichert Paramter auf Flash-EPROM.</USERDESC>
      </DeKOSDESC>
    </F2FUNCTION>
  - <F2FUNCTION DEKOSNR="F2_3" FUNCTYPE="GET" NAME="browseApplicationData">
    - <DeKOSDESC LANGUAGE="dt">
        <SPECDESC>Ähnlich zu browseApplication ermöglicht diese Funktion den Zugriff auf Online-Programm-
          Daten des FG. Dazu zählen das Prozessabbild und andere Speicherbereiche. Dient lediglich der
          Beobachtung.</SPECDESC>
        <ERGONAME>Programmdaten online beobachten</ERGONAME>
        <USERDESC>Liest Prozesswerte aus.</USERDESC>
```

Abbildung 3-4: Ausschnitt aus einer DeKOS Device Description (DDD)

Zur Validierung einer erstellten Beschreibung wurde eine Referenzarchitektur des spezifizierten Bedienmodells als XML-Schema abgebildet [BIRKHOFER 01]. Diese so genannte DRA[37] steht sowohl den Geräteherstellern bei der Entwicklung DeKOS-konformer Feldgeräte als auch dem Endanwender bei der Erweiterung einer Gerätebeschreibung beispielsweise mit einer Komfortfunktion zur Verfügung. Im Rahmen des Verbundprojekts wurden Tools zur Assistenten-gestützten Erstellung und Bearbeitung von Gerätebeschreibungen entwickelt [BENDER ET AL. 02].

[37] DeKOS-Referenzarchitektur

3.3 Industrieumfeld

3.3.1 Proxy-Technologien

In Kapitel 2.1.3 wurde die Proxy-Technologie als Lösung für die Bereitstellung der Hilfsfunktionalität von Automatisierungskomponenten angesprochen. Nachdem Proxy-Lösungen in vielen Bereichen der Automatisierungstechnik ihre Verwendung finden, werden die Mechanismen der damit verbundenen Standardtechnologien nachfolgend näher erläutert. Um den Bedarf einer derartigen Technologie zu begründen, soll nachfolgendes Beispiel äquivalent zu Abbildung 2-1 betrachtet werden:

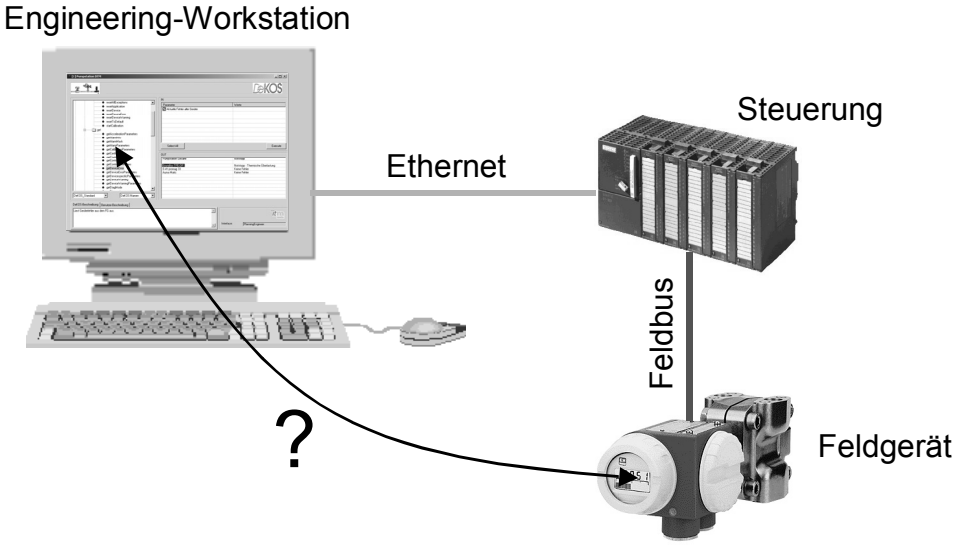


Abbildung 3-5: Zugriff des Bedientools auf ein Feldgerät

Gegeben ist ein Feldgerät zur Durchflussmessung, verbunden über einen Feldbus mit der Steuerung. Diese wiederum kommuniziert über Ethernet mit der Engineering-Workstation (s. Abbildung 3-5). Darauf soll über eine entsprechende Benutzerschnittstelle der Zugriff auf die Hilfsfunktionalität des Feldgerätes ermöglicht werden. Dies könnte über ein proprietäres Bedientool realisiert werden. Allerdings ist dann die Bereitstellung der Funktionalität für andere Applikationen nicht möglich. Daher soll die Software-Architektur des Bedientools derart gestaltet werden, dass die Hilfsfunktionalität des Gerätes über ein Bedienproxy auf der Engineering-Workstation abgebildet wird. Somit könnte sowohl eine Client-Anwendung „Benutzerschnittstelle“ als auch andere Anwendungen auf das Gerät über einheitliche Schnittstellen zugreifen (s. Abbildung

3-6). Eine Zugriffsregelung ist in der Praxis allerdings noch notwendig, auf die aber an späterer Stelle eingegangen werden soll. Der Fokus der Betrachtungen liegt auf der Abbildung des realen Feldgerätes, dem so genannten virtuellen Gerät, softwareseitig realisiert als Stellvertreterobjekt.

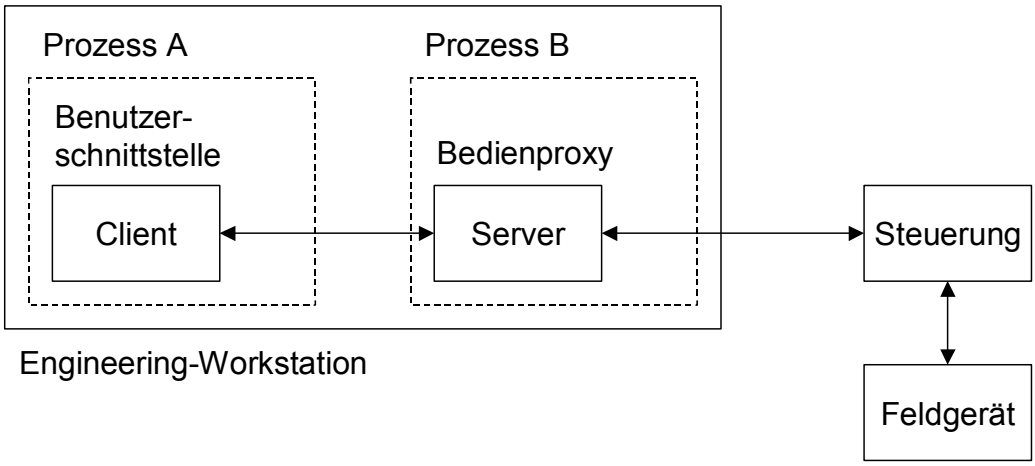


Abbildung 3-6: Zugriff auf Feldgerät über ein Bedienproxy

Dieses Bedienproxy des realen Feldgerätes agiert als Server gegenüber der Benutzerschnittstelle als Client-Anwendung. Die Herausforderungen liegen softwareseitig darin, über die Prozess- und Rechnergrenzen hinweg eine Kommunikation zu ermöglichen. Hierzu verwendet man wiederum ein Stellvertreterobjekt des Servers, das in der Prozessgrenze der Client-Anwendung läuft. Dieses hat zwei entscheidende Eigenschaften:

- Eine identische Schnittstelle wie das eigentliche Serverobjekt
- und eine Referenz auf eben dieses.

Die Referenz enthält dabei alle Informationen die einen erfolgreichen Zugriff auf das Server-Objekt ermöglicht, ist aber meistens kein einfacher Objektzeiger, wie er aus der objektorientierten Programmierung bekannt ist. Das Client-Objekt greift nicht mehr auf das Server-Objekt zu, sondern nur noch auf dessen Stellvertreter, der im lokalen Bereich des Clients abgelegt wird (s. Abbildung 3-7). Insgesamt erreicht man mit der Proxy-Technologie, dass in verteilten Systemen auch entfernte Objekte genauso wie lokale Objekte erscheinen.

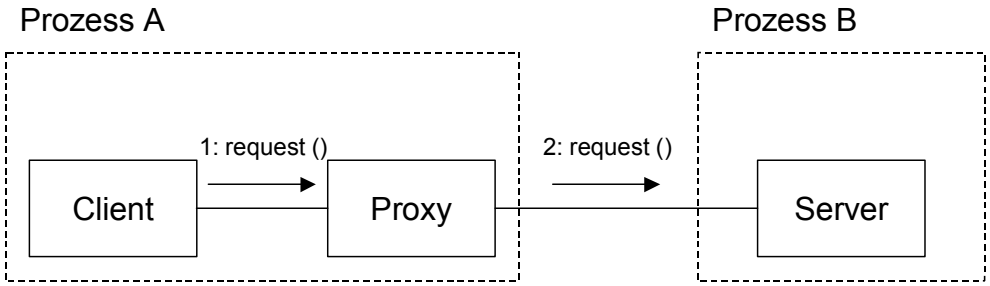


Abbildung 3-7: Prozessübergreifende Kommunikation bei Proxy-Objekten

In den letzten Jahren sind verschiedene so genannte Middleware-Technologien[38] in der Software-Technik entstanden. Als wichtigste Vertreter sind COM (Component Object Model), CORBA (Common Object Request Broker Architecture) und JINI (Java Intelligent Network Infrastructure) zu nennen [MEIER 02]. Als Basis-Technologie hat sich in der Automatisierungstechnik die COM-Technologie etabliert auf die nachfolgend kurz eingegangen wird.

Für Programmiersprachen existiert im Allgemeinen kein Standard der ermöglicht mit Komponenten, die sich in unterschiedlichen Adressräumen befinden, zusammenarbeiten zu können. Erstmals wurde COM von Microsoft mit der Windows Version 3.11 eingeführt und seither weiterentwickelt. Analog zur objektorientierten Programmierung verfolgt COM den Ansatz, Software in Form von wiederverwendbaren Komponenten zu realisieren, die wiederum anderen Komponenten, bzw. Applikationen zur Verfügung stehen. Die Lösung für örtlich verteilte Komponenten, d. h. die Kooperation über Rechnergrenzen hinweg, bietet die später eingeführte Erweiterung DCOM [39].

COM ist dabei keine Strukturierungsvorschrift für Applikationen, sondern stellt vielmehr Technologien zur Verfügung mit deren Hilfe Softwarekomponenten, ganz gleich wo sie lokalisiert sind, zusammenarbeiten können. Selbst unterschiedliche Programmiersprachen sind dabei kein Hindernis, denn COM definiert einen Kommunikationsstandard auf der Ebene von bereits kompilierten, im Speicher abgelegten Code (binäre Ebene) und nicht auf der Basis einer Programmiersprache, wie es bei objektorientierter Programmierung in der Regel der Fall ist. Ein wichtiger Aspekt ist, dass COM-Objekte als vollständig gekapselte Objekte gehandhabt werden. Nach außen hin sind nur die Dienste und Funktionen der Komponente sichtbar, nicht aber die Implementierung [EDDON/EDDON 98].

---

[38] Während im Client vor allem die Schnittstelle zum Benutzer realisiert ist und im Server die Daten bereitgestellt werden, erfüllt die Middleware neben der elementaren Kommunikation unterschiedlichste Dienste, vom Zugriff auf verteilte Daten über Kontrolle verteilter Abläufe bis hin zu verteilter Applikationslogik.

[39] Distributed COM

Die Kommunikation zwischen COM-Objekten erfolgt nach dem Client – Server Prinzip: Eine Softwarekomponente, bzw. Applikation, der Client, fordert Dienste an, die in einer anderen Komponente (dem Server), die außerhalb der Applikation implementiert ist. Beispielsweise handelt es sich um eine Routine, die eine Datei einliest.

In Analogie zu einer objektorientierten Programmiersprache (z. B. C++, Java) behandelt COM einzelne Softwarekomponenten als Klassen. Dies geschieht, wie schon erwähnt, auf binärer Ebene, d. h. unabhängig von einer Programmiersprache bzw. Entwicklungsumgebung. Damit jederzeit eine Lokalisierung mit anschließendem Zugriff auf die Klasse erfolgen kann, werden die einzelnen Klassen von COM anhand ihrer GUID[40] registriert. In der Praxis erfolgt dies durch einen Eintrag in eine zentrale Datei, in der sämtliche auf dem System installierte Applikationen verzeichnet sind. In der Windowswelt wird diese Datenbank als System-Registry bezeichnet. Damit eine Client-Applikation eine bestimmte Komponente nutzen kann, muss ihr die GUID, oder auch CLSID[41] der betreffenden Klasse bekannt sein. Wird die Komponente angefordert, so wird über COM eine Instanz dieser Klasse erzeugt, auf die die Applikation zugreifen kann, diese Instanz wird als eigentliches COM-Objekt bezeichnet.

3.3.2 Komponentenbasierte Automatisierungslösung

Ein etablierter Ansatz zur Unterstützung des Engineerings von Partialsystemen bildet das Automatisierungssystem PROFInet. Dabei wird keinesfalls ein neues Feldbusprotokoll, sondern ein herstellerübergreifender Standard zur anlagenweiten Integration von verteilten automatisierungstechnischen Partialsystemen definiert. Die Technologie bedient sich bereits etablierter Standards der Informationstechnologie.

[40] Globally Unique Identifier - weltweit einmaliger Bezeichner zur Identifikation des Objekts
[41] Class Identifier – spezielle Form einer GUID mit 16 Byte Länge

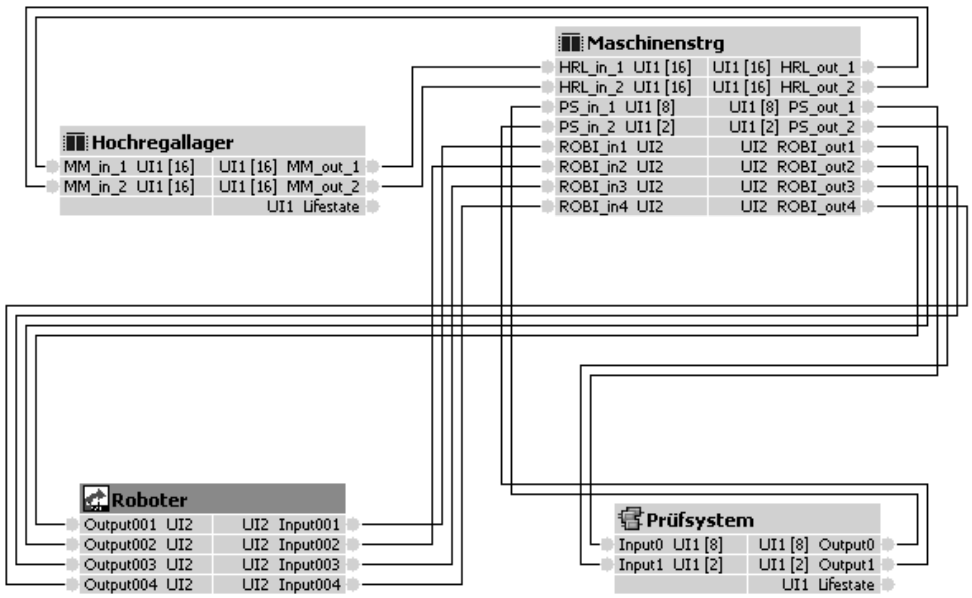


Abbildung 3-8: Die PROFInet-Architektur

PROFInet ermöglicht eine durchgängige Kommunikation zwischen der Feldebene bis zur Ethernet-basierten Leitebene eines Unternehmens und schafft damit eine bisher neue Transparenz im Unternehmen. Dabei kommt es zu einer Steigerung der Flexibilität, da der Anwender mit PROFInet nicht nur auf Komponenten unterschiedlichster Hersteller zurückgreifen kann, sondern in der Lage ist, unterschiedliche Bussysteme miteinander zu kombinieren. zeigt ein PROFInet-System, bei dem zwei Ethernet-basierte Komponenten und zwei Feldbus-Komponenten miteinander vernetzt wurden.

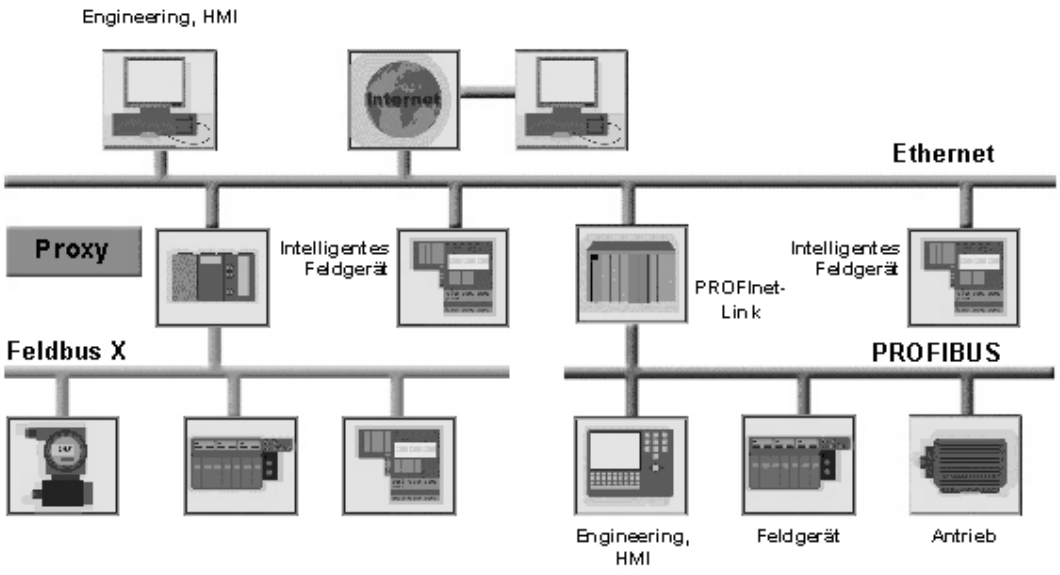


Abbildung 3-9: System mit PROFInet-Kommunikation [PNO 02A]

Durch die Kopplung der Feldebene an übergeordnete Unternehmensebenen entstehen für den Anwender einige wichtige Vorteile. Daten aus der Feldebene, aber auch aus der Sensor- und Aktorebene, können mit PROFInet auf relativ einfache Art und Weise auch in der Office-Welt weiterverwendet werden, beispielsweise in Verbindung mit Wartungs- und Diagnosesystemen. Da als Folge der Globalisierung gerade Tools zur Fernwartung und Ferndiagnose eine immer größere Rolle spielen, gewinnen auch die Möglichkeiten des Internets im Zusammenhang mit PROFInet immer mehr an Bedeutung. Aber auch firmenintern können Vorteile aus der neu entstandenen transparenten Kommunikation gezogen werden. Die Unternehmensleitebene kann schnell und direkt auf Daten der Produktion zugreifen und gegebenenfalls schneller auf besondere Vorkommnisse reagieren.

Dem Entwickler eröffnen sich durch PROFInet weitere Vorteile. Die horizontale Integration neuer Module war bereits durch bestehende Feldbussysteme gelöst, PROFInet eröffnet aber ganz neue Wege im Bereich der vertikalen Integration.

Der Ablauf des Engineerings nach der Erstellung der PROFInet-Komponente ist in Abbildung 3-10 dargestellt. Zunächst werden die Beschreibungen der PROFInet-Komponenten in einen so genannten Verschaltungseditor geladen. Die einzelnen Komponenten werden dann ohne Programmierkenntnisse miteinander verknüpft. Anschließend wird die Verschaltungsinformation in die einzelnen Komponenten geladen.

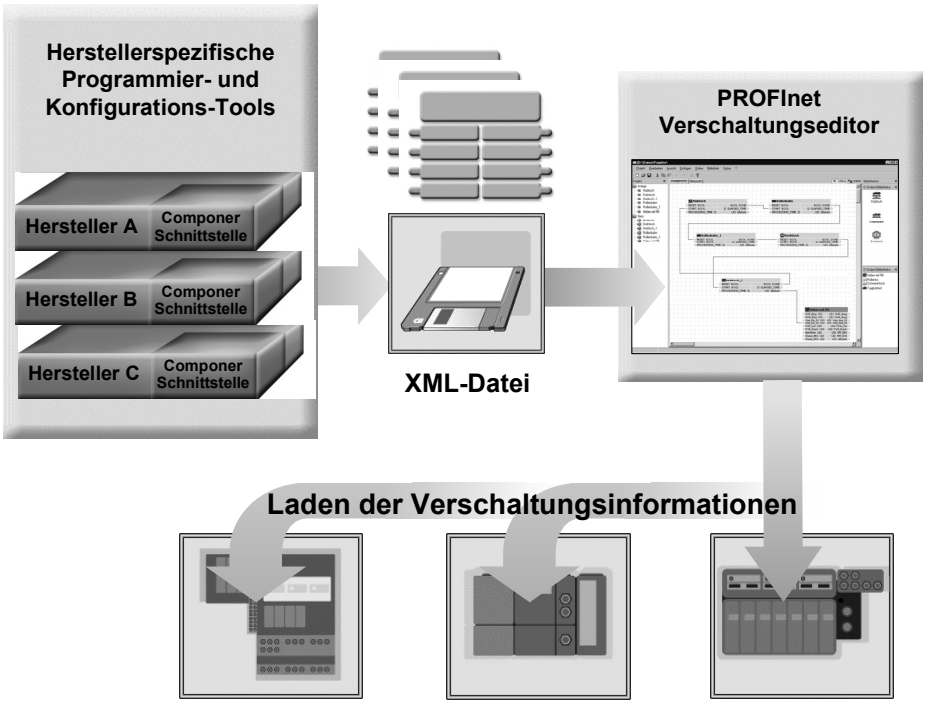


Abbildung 3-10: Engineeringablauf bei PROFInet [PNO 02A]

Wie aus Abbildung 3-10 ersichtlich, setzt PROFInet auf herstellerspezifische, sprich proprietäre Tools zur Bedienung der Komponenten. Eine einheitliche, zentrale Schnittstelle zur Bedienung des Partialsystems ist derzeit nicht realisiert.

3.3.3 Asset Management

In der Prozessleittechnik gewinnen, neben den klassischen Aufgaben wie Prozessüberwachung und Prozessführung, Funktionen zur Verwaltung und Optimierung der Feldgeräte und Anlagenkomponenten eines Produktionsbetriebes zunehmend an Bedeutung [NICKLAUS/FUß 00, NE 91]. In der industriellen Praxis lässt sich jedoch beobachten, dass die Möglichkeiten neuer intelligenter Feldgeräte in Bezug auf deren Diagnose- und Instandhaltungsinformationen oft nur unvollständig oder gar nicht genutzt werden [MÜLLER ET AL. 02].

An dieser Stelle setzt das Konzept der ACPLT/AMBox[42] an. Hierdurch ist ein offener, konfigurationsfreier und herstellerunabhängiger Zugang zu den Feldgeräten eines Partialsystems möglich. Abbildung 3-11 zeigt die Integration der AMBox parallel zu einem bestehenden System.

[42] Aachener Prozessleittechnik/Asset Management Box

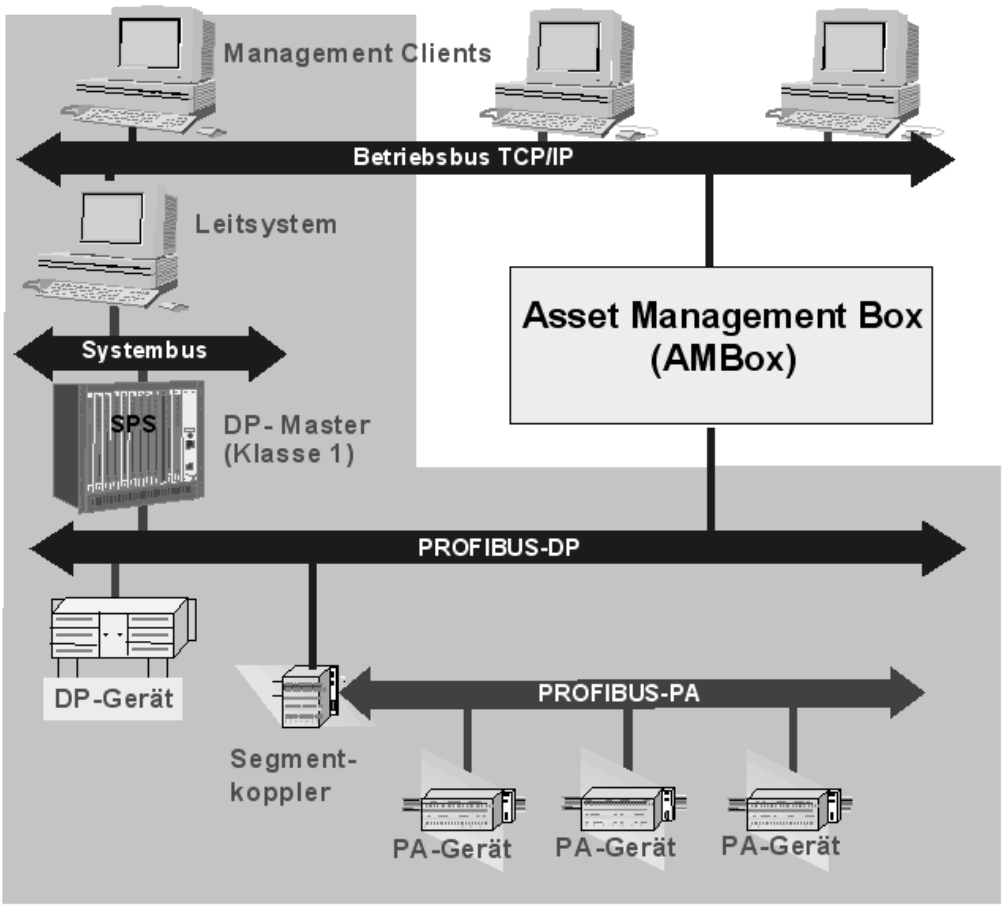


Abbildung 3-11: Integration der ACPLT/AMBox [ENSTE/HÜBNER 03]

Die AMBox ist ein Industrie-PC, der als Profibus-DP Master Klasse 2 im Feldverbund agiert. Dabei wird nach Start selbstständig ein komplettes Geräteabbild aus bis zu 6 Profibus-PA Feldbussegmenten aufgebaut. Die gesammelten Geräteinformationen werden in einem offenen Objektverwaltungssystem strukturiert und frei zugänglich hinterlegt. Mit Hilfe von Standardanwendungsmodulen oder frei programmierbaren Modulen können diese Informationen in vielfältiger Weise genutzt werden. Nachfolgende Standardfunktionen werden u. a. vom System angeboten:

- Rückdokumentation des Partialsystems im XML-Format
- Instandhaltungs- und Inbetriebnahmefunktionen
- Fernwartungs- und Diagnosemöglichkeit
- Automatische Parametrierung bei Gerätetausch
- Geräteverfolgung über den ganzen Lifecycle eines Gerätes

Es existieren weitere Ansätze von Asset Management Lösungen von E+H [E+H 03], AMS von Fisher-Rosemount [BECKER 01] etc. die im Einklang mit

der Namur NE 91 Hilfsfunktionalität auf einer höheren Abstraktionsebene anbieten. Alle Ansätze haben gemeinsam, dass sie keine einheitlichen Hilfsfunktionen anbieten. Darüber hinaus fehlt ein Modell zur Abbildung der Funktionalität; eine Erweiterung der Systeme mit anderen Herstellern ist nicht möglich. Die zusätzliche Funktionalität beschränkt sich hauptsächlich auf die Diagnose. Der Mehrwert an Funktionalität wird geschmälert durch die fehlende Kapselung.

3.3.4 Konzepte zur Feldgerätebedienung

In den letzten Jahren haben sich verschiedene Technologien zur Gerätebedienung etabliert. Die zwei wichtigsten Vertreter sollen nachfolgend kurz beschrieben werden. Beiden Technologien ist gemeinsam, dass der Anwender ein spezielles Computerprogramm auf seiner Engineering-Workstation benötigt. Des Weiteren gehen beide Systemarchitekturen nur auf das Engineering einzelner Geräte ein. Die Bedienung von Partialsystemen findet keine Berücksichtigung.

Electronic Device Description (EDD)

Mit der EDD wird ein Feldgerät in seinen gerätespezifischen Eigenschaften beschrieben. Enthalten sind dabei Informationen über den Hersteller, den Gerätetyp, die Geräteversion, die Beschreibung der Benutzeroberfläche, der Parameter und der Kommunikation. Die Gerätebeschreibung basiert dabei auf der EDDL[43], einer textbasierenden Beschreibungssprache. Eine EDD-Applikation, wie beispielsweise SIMATIC PDM, Commuwin, etc., interpretiert die EDD eines Gerätes und erzeugt die Benutzeroberfläche für den Anwender. Die Erstellung der EDD erfolgt durch den Gerätehersteller oder einen Dienstleister und wird gewöhnlich zusammen mit dem Gerät ausgeliefert [PNO 02B].

[43] Electronic Device Description Language

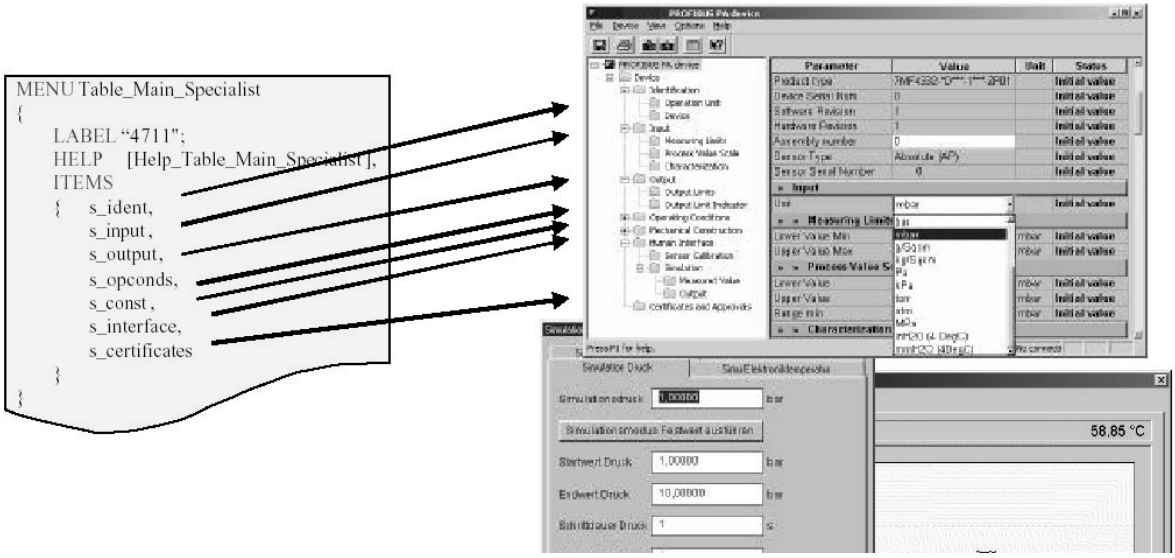


Abbildung 3-12: Generierung der Benutzeroberfläche aus der EDD [PNO 02B]

Durch die EDD-Applikation können alle notwendigen Engineering-Aufgaben ausgeführt werden. Hierzu zählen neben Inbetriebnahme, Wartung und Diagnose auch Funktionen zur Hilfe und Dokumentation. Komfortable Funktionen, die das Wartungspersonal beim Austausch von Geräten unterstützen, sind genauso realisierbar wie rollenorientierte Sichten auf die Gerätefunktionalität über Benutzerprofile.

Field Device Tool (FDT)

Die FDT-Technologie spezifiziert ein Schnittstellenkonzept um Bedienproxys für Feldgeräte, die als DTM[44] bezeichnet werden, in unterschiedlichen Applikationen beliebiger Hersteller nutzen zu können. Dabei sei darauf hingewiesen, dass die Technologie kommunikationsunabhängig für beliebige Feldbussysteme einsetzbar ist. Die DTMs basieren auf der Microsoft COM-Technologie, welche in der Windows-Welt seit Jahren etabliert ist. COM-basierte Komponenten können ihre Funktionalität nach außen hin für andere Applikationen zur Verfügung stellen. Demzufolge werden DTMs vom Hersteller des jeweiligen Feldgerätes, wie dies auch bei der GSD[45] oder EDD der Fall ist, mit ausgeliefert und stellen ihre Funktionalität in einer standardisierten, FDT-fähigen Rahmenapplikation zur Verfügung. Der Proxy dient dann dem Engineering eines Feldgerätes über den gesamten Lifecycle hinweg von der Inbetriebnahme bis zur Diagnose im Runtime-Modus. Dabei kennt das DTM alle Regeln des Gerätes, wie z. B. Plausibilisierung, und stellt alle Benutzerdialoge über ein eigenes GUI bereit.

---

[44] Device Type Manager
[45] Gerätestammdaten

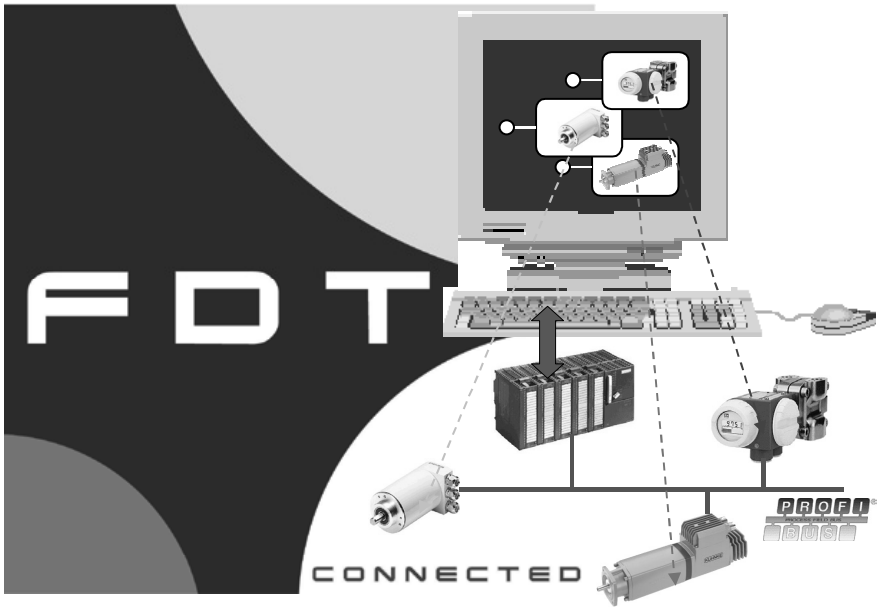


Abbildung 3-13: Field Device Tool (FDT)

Neben den Geräte-DTMs gibt es DTMs für die Kommunikation mit dem realen Feldgerät, beispielsweise über Profibus-DP oder HART[46]. Die Verknüpfung, Verwaltung und Zuordnung der Geräteproxys zum jeweiligen Kommunikations-DTM übernimmt die übergeordnete Software, in der die DTMs unter Nutzung der COM-Technologie eingebettet sind, im Allgemeinen als FDT-Frameapplication bezeichnet.

FDT ist, wie zuvor beschrieben, noch keine implementierte Software, sondern vielmehr die Spezifikation, auf deren Basis ein beliebiges Engineeringtool durch Implementierung der spezifizierten Schnittstelle FDT-tauglich gemacht werden kann. Die Integration dieser spezifizierten Schnittstelle ermöglicht die Nutzung von DTMs. Auf dem Markt gibt es derzeit bereits verschiedene Engineering-tools die die FDT-Schnittstelle unterstützen, beispielsweise die Prozessleitsys-teme Freelance oder Symphonie von ABB. Zusätzlich stehen FDT-fähige Stand Alone Applikationen, wie z. B. PACTware zur Verfügung, in deren Umgebung DTMs einfach per Plug and Play integriert werden können [MEYER 01].

[46] Highway Addressable Remote Transmitter

3.4 Zusammenfassung

3.4.1 Bewertung

Die in diesem Kapitel vorgestellten Konzepte und Systemarchitekturen zur Bedienung von intelligenten Feldgeräten und mechatronischen Systemen werden nachfolgend gemäß der *Definition 2.7* und den Anforderungen aus Kapitel 2.4 beurteilt. Die Tabelle gibt einen Überblick der angesetzten Kriterien:

Bewertungskriterium:	DIN 13306	DIN 31051	VDI 2186	VDI 2187	VDI 3850	NE 57	NE 91	DeKOS
Relevant für die Bedienung von Partialsystemen	1	1	0	1	1	0	1	1
Relevant für die Bedienung von einzelnen Feldgeräten	1	1	1	1	0	1	1	1
Berücksichtigung der Geräteabhängigkeiten	0	0	0	0	0	0	1	0
Standardisierte Hilfsfunktionen	0	0	0	0	0	0	0	1
Einheitliches Look&Feel des Bedienobjekts	0	0	1	1	1	1	0	1
Berücksichtigung eines Rollenmodells	0	0	0	1	0	0	1	1
Strukturierung der Hilfsfunktionen	0	1	0	0	0	1	0	0
Berücksichtigung der Inbetriebnahme	0	0	1	0	0	1	1	1
Berücksichtigung der Instandhaltung	1	1	1	0	0	1	1	1
Ansatz bezogen auf Verfahrenstechnik	1	1	0	1	0	0	1	1
Ansatz bezogen auf Fertigungstechnik	1	1	1	1	1	1	0	1
Summe (von max. 11):	5	6	5	6	3	6	7	9

Tabelle 3-3: Kriterien zur Beurteilung der Eignung der vorgestellten Konzepte

Anhand der Tabelle 3-3 ist ersichtlich, dass mit keinem derzeitigen Engineering-Konzepte eine vollständige Lösung des in Kapitel 2.4 identifizierten Problems möglich ist. Jedoch hat die Analyse ergeben, dass auf Basis des DeKOS-Bedienmodells, unter Einbeziehung der relevanten Konzepte eine Lösung erzielt

werden kann. Die Relevanz der einzelnen Ansätze soll im Folgenden erläutert werden:

- In der DIN 13306 und DIN 31051 werden Instandhaltungsmaßnahmen beschrieben, die auch für die Bedienung der Partialsysteme von Bedeutung sind. Des Weiteren werden Begriffe definiert, die zur Benennung der Hilfsfunktionen auf Partialsystemebene herangezogen werden können.
- Die VDI 2186 beschränkt sich zwar auf den Einsatz von Frequenzumrichtern, enthält aber wichtige Ansätze zur Systematisierung der angebotenen Hilfsfunktionalität.
- Die Rollenmodelle und Gestaltungshinweise aus der VDI 2187 sind allgemeingültig und somit auch bei einer Partialsystembedienung zu berücksichtigen.
- Die VDI 3850 kann bei der Lösung des Problems vernachlässigt werden, da hierbei nur Ansätze enthalten sind, die sich auch in anderen Schriften wiederfinden.
- Genauso wie die VDI 2186 beschränkt sich die NE 57 auf den Einsatz von Frequenzumrichtern, beschreibt aber auch die Hilfsfunktionalität systematisiert.
- Die NE 91 enthält Anforderungen an die Bedienung von Partialsystemen und die damit verbundenen zu berücksichtigenden Abhängigkeiten der Geräte.
- Ausschließlich das DeKOS-Bedienmodell enthält eine Bibliothek von standardisierten Hilfsfunktionen, die unabhängig vom Hersteller und der Geräteklasse eingesetzt werden können.

Die vorgestellten Technologien die derzeit zur Bedienung von Feldgeräten zur Verfügung stehen, sollen ebenfalls in Bezug auf ihre Eignung zur Bedienung von mechatronischen Systemen bewertet werden. Dabei wird auf eine tabellarische Darstellung verzichtet:

- Die derzeit einzige komponentenbasierte Automatisierungslösung ist die PROFInet-Technologie. Allerdings beschränkt sie sich in der aktuellen Version auf die Prozessfunktion eines Partialsystems. Für die Bedienung der Partialsysteme muss auf die proprietären Lösungen der unterlagerten Geräte zurückgegriffen werden.
- Die AMBox bietet zwar einen konfigurationsfreien Zugang zu den Gerätedaten eines Partialsystems an, beschränkt sich aber lediglich auf Geräte der Verfahrenstechnik mit Profibus-PA Busanschaltung. Des Weiteren wird die Hilfsfunktionalität nicht standardisiert angeboten. Ansätze, wie beispielsweise der Austausch eines Feldgerätes des Partialsystems, ohne

Kenntnis der Interna, erfüllen teilweise die Anforderungen aus Kapitel 2.4.

- Auf Basis der heutigen EDD-Tools ist ausschließlich eine Bedienung von einzelnen Geräten möglich.
- Mit der FDT-Technologie ist ebenfalls nur eine Bedienung von einzelnen Feldgeräten angedacht. Eine Strukturierungsvorschrift für die angebotenen Funktionen ist enthalten.

3.4.2 Fazit und Lösungsansatz

Die dargestellten Modelle und Architekturen beschäftigen sich hauptsächlich mit der Gerätebedienung. Dabei wird größtenteils noch mit einer datenorientierten Sicht auf die Gerätefunktionalität gearbeitet. Funktionsorientierte Ansätze, wie bei FDT möglich, unterliegen keinem Standard, sodass auch hier eine einheitliche Bedienung nicht gegeben ist.

Forderungen nach einer möglichen Bedienung von Partialsystemen, wie beispielsweise in der NE 91, werden nur unzureichend von den derzeitigen Technologien berücksichtigt. Eine einheitliche Lösung sowohl für verfahrens- als auch fertigungstechnische Anlagen ist nicht vorhanden.

Zusammenfassend geht aus der Bewertung der Engineering-Konzepte und Technologien hervor, dass mit herkömmlichen Mitteln die Anforderungen zur Kompensierung des beschriebenen Defizits aus Kapitel 2 nicht gelöst werden können. Allerdings sind verwertbare Ansätze vorhanden, die in den Lösungsansatz mit einfließen werden. Ansätze zur einheitlichen Bedienung von intelligenten Feldgeräten wie die Idee des DeKOS-Bedienmodells sollen bei der Lösungsfindung zur Bedienung von mechatronischen Systemen einbezogen werden. Schließlich soll der Bediener keine neue Bediensicht auf Verbundebene vorfinden, sondern Partialsysteme genauso wie einzelne Geräte intuitiv bedienen können.

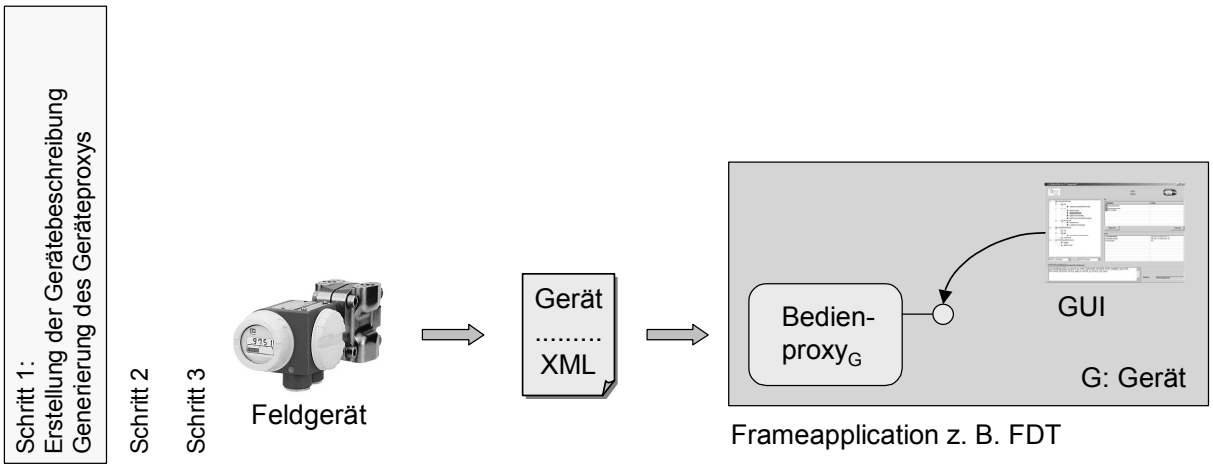


Abbildung 3-14: Generierung eines Bedienobjekts auf Geräteebene

Der Lösungsansatz gestaltet sich dreiteilig. Das zuvor beschriebene DeKOS-Bedienmodell scheint grundlegend geeignet, um die Hilfsfunktionalität der Geräte und Partialsysteme standardisiert abbilden zu können. Eine Analyse repräsentativer Partialsysteme soll hierzu Aufschluss geben. Allerdings ist zu untersuchen, inwieweit eine Grundfunktionalität von den Geräten zu erbringen ist. Durch die Abbildung der Geräte über Proxy-Technologien wie beispielsweise FDT steht die Funktionalität aller Geräte eines Systems auf der Engineering-Workstation zur Verfügung (s. Abbildung 3-14). Um allerdings nicht nur über die Benutzerschnittstelle des DTMs die Hilfsfunktionen der Geräte nutzen zu können, sondern auch anderen Software-Objekten zur Verfügung zu stellen, ist die Software-Architektur zu erweitern.

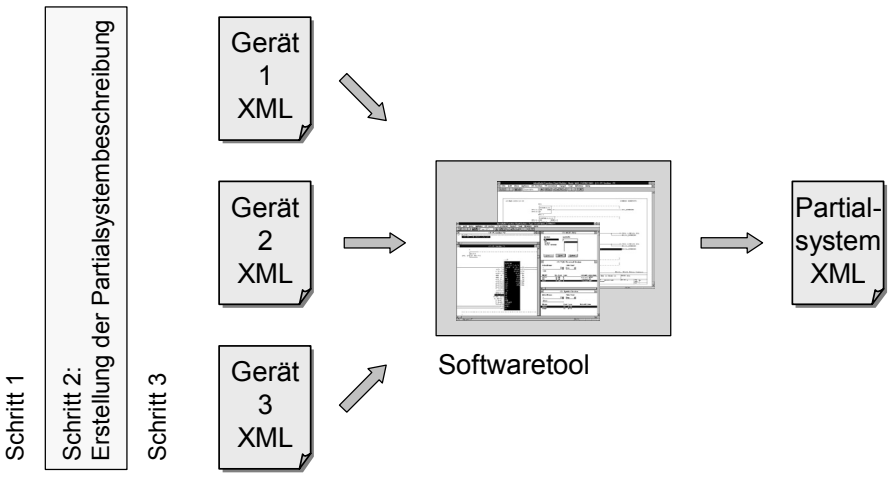


Abbildung 3-15: Generierung eines Bedienobjekts auf Partialsystemebene

Es ist ein Vorgehensmodell zu entwickeln, das auf Basis der standardisierten Gerätebeschreibungen eines Partialsystems eine assistentengestützte Generierung der Partialsystembeschreibung ermöglicht (s. Abbildung 3-15). Hieraus ist wiederum äquivalent zum Feldgerät eine Generierung des Bedienproxy durchführbar (s. Abbildung 3-16). Dieses stellt die Hilfsfunktionalität des Partialsystems dem Bediener über eine einheitliche Benutzerschnittstelle zur Verfügung.

Dabei greift das Bedienproxy nicht direkt auf die unterlagerten Geräte zu, sondern macht sich die bereitgestellte Funktionalität der einzelnen Bedienproxys der Geräte zu Nutze, um eine vom Bediener aufgerufene Funktion auszuführen. Die Abhängigkeiten zwischen den Geräten bleiben somit dem Bediener verborgen.

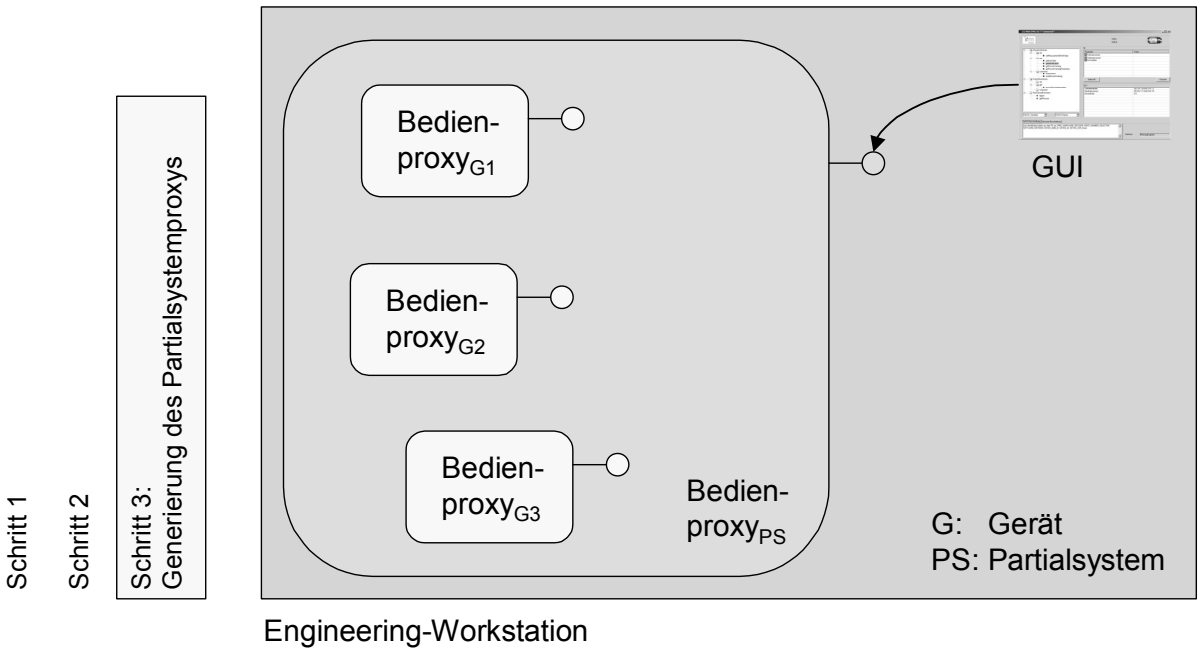


Abbildung 3-16: Kapselung der Gerätefunktionalität

Im nachfolgenden Kapitel wird das Lösungskonzept erarbeitet. Hierzu folgt nach einer detaillierten Betrachtung des Lifecycles von Partialsystemen ein Beispiel aus der Verfahrenstechnik. Nach dem Beleg des Bedarfs nach Modellierung und Beschreibung folgt die Spezifikation der erforderlichen Klassenmodelle zur Abbildung der Hilfsfunktionalität von Teilanlagen.

4 Lösungskonzept

Die bisherigen Ausführungen haben ergeben, dass sowohl Hersteller von Feldgeräten als auch die Anwender der Geräte mit einer sehr großen Vielfalt an Hilfsfunktionalität und Anzahl an Feldgeräten verbunden mit den Abhängigkeiten konfrontiert sind. Lösungen wie das DeKOS-Bedienmodell bieten eine gute Basis, um durch eine standardisierte Hilfsfunktionalität den Nutzen für den Hersteller von Feldgeräten als auch für den Endanwender auf Systemebene steigern zu können.

Ansätze zur Systembedienung die sowohl dem Hersteller als auch Anwender von Nutzen sind, finden sich ansatzweise in den zuvor beschriebenen Richtlinien und Empfehlungen die allerdings bisher industriell nur unzureichend umgesetzt sind. Auch die Einführung neuer Technologien wie PROFInet oder die AMBox lösen das Problem der Systembedienung nur unzureichend oder gehen proprietäre Wege, die wiederum das Ziel einer einheitlichen Lösung verfehlen.

Das nachfolgende Kapitel beschäftigt sich ausführlich mit dem Lösungskonzept des zuvor identifizierten Problems. Nach einer Analyse der mechatronischen Systeme bezüglich des Lifecycles und des Nutzens der Hilfsfunktionalität auf Systemebene ist ein ausführliches Beispiel gegeben. Der Bedarf nach Klassifizierung der Hilfsfunktionalität, Modellierung und Beschreibung als Basis für ein Vorgehensmodell zur Generierung von Systembedientools wird belegt. Abschließend folgt die Beschreibung der Defizite der bestehenden DeKOS-Referenzarchitektur. Die mögliche Lösung ist zu diskutieren und detailliert zu beschreiben.

4.1 Analyse mechatronischer Systeme

4.1.1 Lifecycle der Partialsysteme

In Kapitel 2.2.2 wurde auf den allgemeinen Lebenszyklus eines Produkts von der Idee bis zur Entsorgung eingegangen. Dabei wurde erstmalig ein Ansatz zur Ordnung der Begriffswelt im Bereich der Produktnutzung gegeben (s. Abbildung 2-5). Basierend auf dem vorgestellten allgemeingültigen Modell wird nachfolgend der Lifecycle von Partialsystemen bezüglich deren Hilfsfunktionalität beschrieben und auf die damit verbundene Bedienung näher eingegangen (s. Abbildung 4-1).

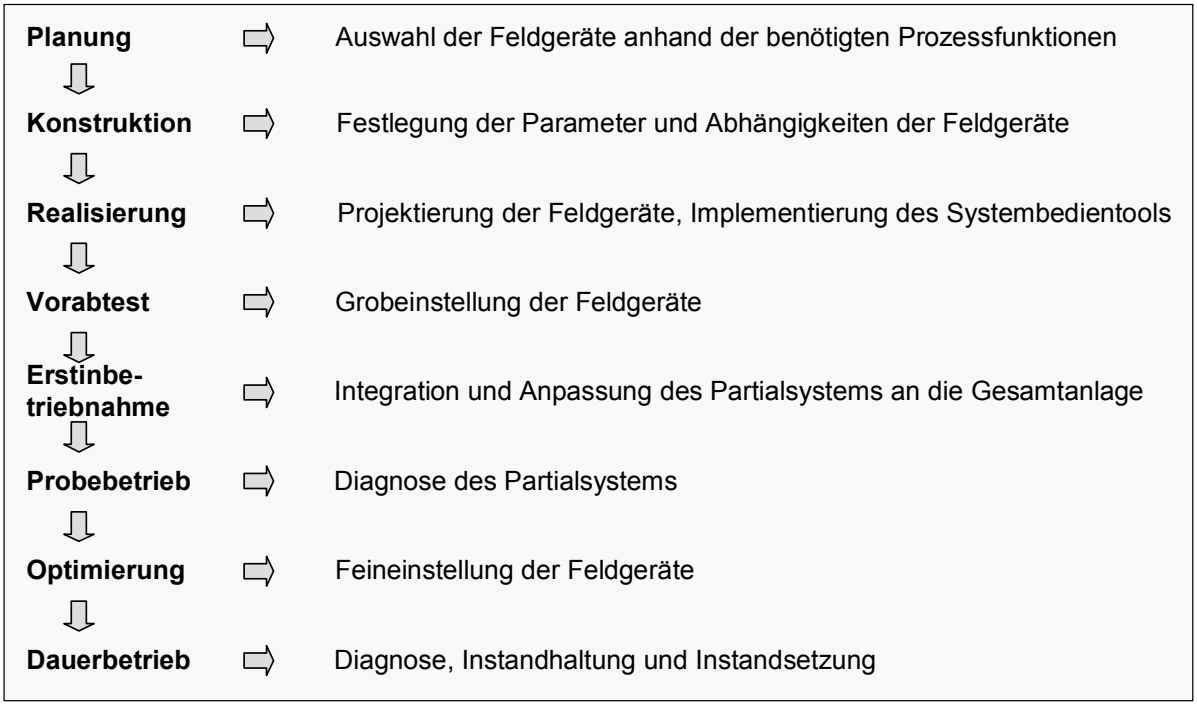


Abbildung 4-1: Lifecycle von Partialsystemen bezüglich der Hilfsfunktionalität

Partialsysteme entstehen bei der Planung einer Anlage. Ein spezifizierter Teilprozess ist durch die Auswahl geeigneter Feldgeräte mit bestimmten Prozessfunktionen zu realisieren. Neben der mechanischen Konstruktion sind im nächsten Schritt die Sollparameter und Abhängigkeiten der eingesetzten Geräte zu spezifizieren. Bei der Realisierung projektiert der Anlagenbauer die einzelnen Geräte, d. h. es erfolgt die Grundparametrierung. Neben dem Steuerungsprogramm entsteht die Visualisierung. Diese enthält hauptsächlich bei verfahrenstechnischen Anlagen neben dem reinen Prozessbedienen und -beobachten Softwarekomponenten zur Gerätebedienung und eventuell zusätzlich Asset Management Tools (s. Kapitel 3.3.3). Sofern möglich erfolgt noch beim Anlagenbauer ein Vorabtest des Partialsystems. Bei der Erstinbetriebnahme beim Kunden wird die Teilanlage in das bestehende System integriert oder sofern es sich um eine komplett neue Anlage handelt, mit anderen Partialsystemen zusammengeschlossen. Bei einem Probebetrieb ist eine Diagnose des Partialsystems möglich. Eventuell auftretende Fehler können rechtzeitig erkannt und behoben werden. Bei der Optimierung des Prozesses ist eventuell noch einmal auf die Hilfsfunktionalität der einzelnen Feldgeräte zuzugreifen, um eine Feineinstellung vorzunehmen. Sind alle Arbeitsschritte erfolgreich vollzogen, kann der Dauerbetrieb beginnen.

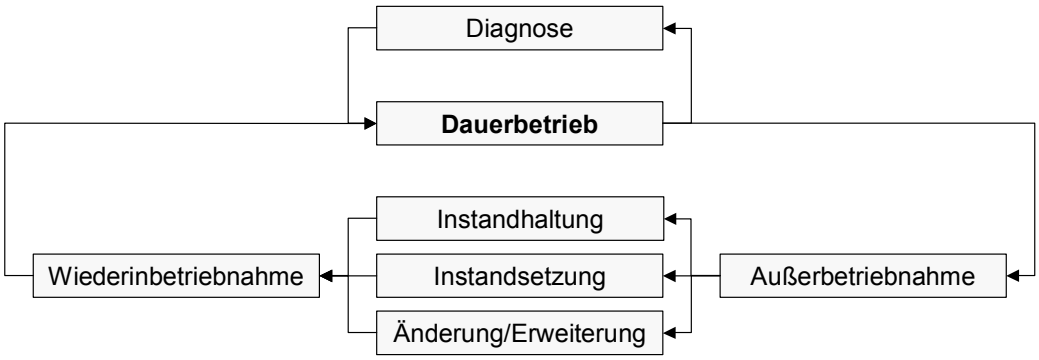


Abbildung 4-2: Anlagenbedienung während des Dauerbetriebs

Während des Dauerbetriebs sind Diagnosemaßnahmen am Partialsystem durchführbar. Hierdurch ist die Funktionsfähigkeit der Teilanlage nicht beeinträchtigt (s. Abbildung 4-2). Sind darüber hinaus Maßnahmen zur Instandhaltung, beispielsweise die Kalibrierung, zur Instandsetzung, wie der Austausch eines Frequenzumrichters oder Änderungen und Erweiterungen durchzuführen, ist das Partialsystem außer Betrieb und nach Durchführung der Arbeit wieder in Betrieb zu nehmen.

Der Rückbau ist die letzte Phase im Lebenszyklus eines Partialsystems. Wird dieses nicht mehr benötigt, ist es nicht mehr rentabel änderbar oder erweiterbar, erfolgt die Entsorgung. Eventuell können Komponenten des Partialsystems durch Recycling bei der Realisierung einer neuen Anlage Wiederverwendung finden.

4.1.2 Nutzen der Hilfsfunktionalität auf Systemebene

Die Handhabung der mechatronischen Systeme soll für den Endanwender, in der Rolle des Anlagenbetreibers, vereinfacht werden. Es ergeben sich drei Gruppen von Akteuren[47], die bei der Realisierung von Bedientools auf Systemebene in den Entstehungsprozess und bei der Benutzung dieser involviert sind (s. Abbildung 4-3).

Es folgt eine konkrete Betrachtung der Anforderungen an ein Systembedientool. Dabei finden die Ansätze zum anlagennahen Asset Management in [NE 91], [KOSCHEL/MÜLLER/NICKLAUS 00], [NICKLAUS/FUSS 00] Berücksichtigung.

[47] Auch als Benutzergruppe oder -rolle bezeichnet [VDI 2187].

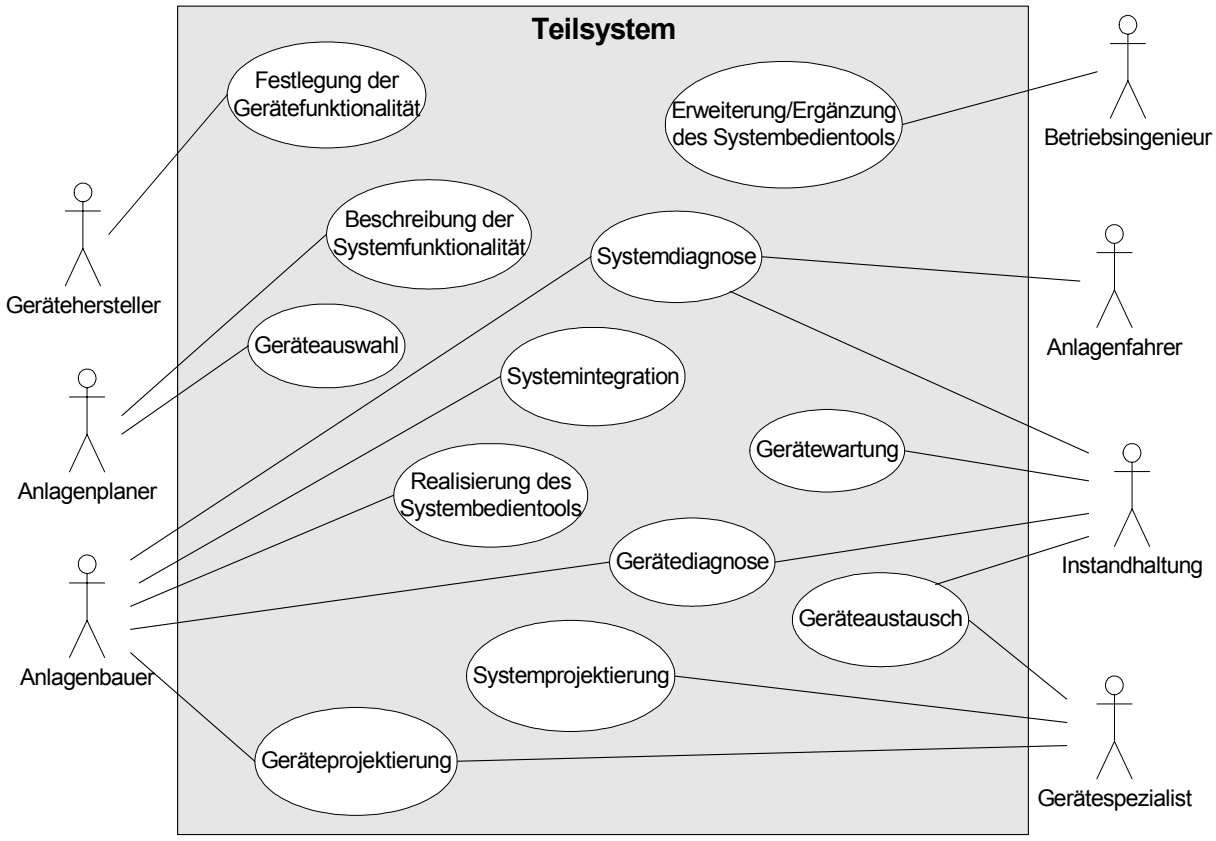


Abbildung 4-3: Anwendungsfalldiagramm

Als erstes ist der Anlagenbetreiber zu nennen. Dieser hat in Anlehnung an die Richtlinien VDI 2187 und NE 91 nachfolgende Unterakteure:

- Anlagenfahrer
- Instandhaltung
- Gerätespezialist
- Betriebsingenieur

Der Anlagenfahrer wird sich hauptsächlich der Prozessführung widmen. Allerdings muss er in der Lage sein, anhand einer Systemdiagnose und der daraus resultierenden Fehlermeldungen Entscheidungen zu treffen, wie beispielsweise bei einer auftretenden Störung zu verfahren ist, um diese möglichst schnell zu beseitigen.

Die innerbetriebliche Instandhaltung wird auch zuerst eine Systemdiagnose durchführen, um eine Übersicht des Zustands der Teilanlage zu bekommen. Es muss der Instandhaltung möglich sein, sich bei Diagnose- und Fehlermeldungen vom Gesamtsystem zum einzelnen Feldgerät informieren zu können. Zusätzliche Maßnahmen wie die Kalibrierung von Sensoren sind durch das Systembedientool zu unterstützen. So wäre es denkbar, dass der Instandhaltungsmitarbeiter zur täglichen Kalibrierung von pH-Sensoren unabhängig von Gerät und Hersteller einen Zugangspunkt in seiner Benutzeroberfläche hat, die ihm die not-

wendige Hilfsfunktionalität zur Durchführung der notwendigen Arbeiten an allen Geräten anbietet. Zusätzlich sollte es der Instandhaltung möglich sein, einen Geräteaustausch durchzuführen. Hierzu ist es erforderlich, dass alle notwendigen Projektierungsdaten des Gerätes dem Systembedientool durch eine entsprechende Datenbank zur Verfügung stehen, sodass dem Mitarbeiter keine weiteren Unterlagen, Disketten o. ä. zur gestellt werden müssen. Er benötigt auch kein Spezialwissen der Geräteklasse oder des Herstellers. Ein Softwareassistent leitet durch den Geräteaustausch.

Sind Änderungen an den Projektierungsdaten, der Firmware etc. durchzuführen, ist der Gerätespezialist zuständig. Er ist Systemexperte und verfügt über das Detailwissen aller Geräte, Apparate und Maschinen. Änderungen die das ganze Partialsystem betreffen, wie an der Anfahrrampe eines Transportbandes bestehend aus mehreren Antrieben, erfolgen gleichzeitig durch eine Systemprojektierung. Er hat das ausschließliche Recht, die Parametersätze der Datenbank zu ändern.

Die Funktionalität, die das Systembedientool anbietet, ist durch den Anlagenplaner spezifiziert. Der Anlagenbauer wiederum realisiert dieses nach der Spezifikation. Das Systembedientool muss allerdings derart gestaltet sein, dass der Betriebsingenieur ohne großen Aufwand Erweiterungen und Ergänzungen an dem bestehenden System vornehmen kann. Hierzu ist er beim Engineering dahingehend zu unterstützen, dass die erforderlichen Maßnahmen ohne Programmierungsaufwand durchzuführen sind.

Bei dem Entstehungsprozess einer Anlage von der Planung bis zum Dauerbetrieb (s. Abbildung 4-1) sind darüber hinaus beteiligt:

- Anlagenplaner
- Gerätehersteller
- Anlagenbauer

Die Spezifikation der gesamten Anlage erfolgt durch den Anlagenplaner. Dieser kann auch Mitarbeiter des Anlagenbetreibers sein. Durch ihn erfolgt anhand des geforderten Prozesses eine Beschreibung der Systemfunktionalität. Diese hat standardisiert zu erfolgen, um u. a. wie zuvor beschrieben dem Betriebsingenieur einfache Änderungen und Erweiterungen zu ermöglichen. Bei der Planung erfolgt auch die Auswahl der Feldgeräte. Dabei wäre es wünschenswert, wenn der Anlagenplaner keiner Einschränkung durch Gerätehersteller etc. unterworfen ist.

Die Geräte sind durch den Hersteller derart zu gestalten, dass sie die benötigte Funktionalität zur Realisierung der Systembedientools enthalten. Der Hersteller ist bei der Auswahl der Funktionalität seiner Feldgeräte in geeigneter Weise zu unterstützen.

Die Realisierung der Anlage erfolgt durch den Anlagenbauer. Neben dem mechanischen Aufbau der Komponenten und der Implementierung des Steuerungsprogramms erfolgt durch ihn die Integration der Feldgeräte in die Anlage. Dabei wird die Gerätegrundprojektierung durchgeführt. Anhand der standardisierten Beschreibung der Systemfunktionalität der geforderten Hilfsfunktionen wird das Systembedientool realisiert. Der Engineering-Aufwand ist durch die Unterstützung durch Softwaretools zu minimieren. Beim Vorabtest und der Integration des Partialsystems in die Gesamtanlage erfolgt die Systemdiagnose und eventuell eine Nachprojektierung der Feldgeräte um den Herstellungsprozess zu optimieren und aufgetretene Fehler zu beseitigen.

4.1.3 Beispiel der Hilfsfunktionalität eines Wärmetauschers

Wie später noch genauer untersucht wird dienen Hilfsfunktionen auf Systemebene nicht nur dazu Funktionalität der unterlagerten Geräte dem Bediener zusammengefasst anzubieten (s. Kapitel 2.2.4). Wie MEYER schon in [MEYER/BENDER 01] und [MEYER/BREGULLA/BENDER 01A] untersucht hat, können auch Anlagenteile wie Rohrleitungen, Kessel etc., die von sich aus keine Hilfsfunktionalität aufweisen, durch den sinnvollen Verbund von Feldgeräten diagnostiziert werden. Dieser „Mehrwert“ an Funktionalität ist auch Bestandteil des Asset Managements (s. Kapitel 3.3.3) und ist durch einen Kreativprozess bei der Anlagenplanung zu berücksichtigen.

Bei allen verfahrenstechnischen Prozessen spielt die Energieübertragung für die Reaktionsführung eine große Rolle. In den meisten Fällen handelt es sich hierbei um Wärme, die innerhalb und zwischen verschiedenen Systemen übertragen wird. Eine genaue Kenntnis der Wärmeübertragungsmechanismen ist unumgänglich, um einen sicheren Ablauf des Prozesses und eine richtige Dimensionierung der Reaktoren, Wärmetauscher und Rohrleitungen zu gewährleisten. Neben chemischen und verfahrenstechnischen sind auch wirtschaftliche Gesichtspunkte von großer Bedeutung, da die Energiekosten bei der Durchführung eines Prozesses möglichst klein zu halten sind.

Das nachfolgende Beispiel erläutert die Hilfsfunktionalität eines Wärmetauschers. Dabei wird gezeigt, dass dieses Partialsystem nicht nur Diagnosefunktionen aufweist, sondern auch über Wartungs- und Projektierungsfunktionen verfügt.

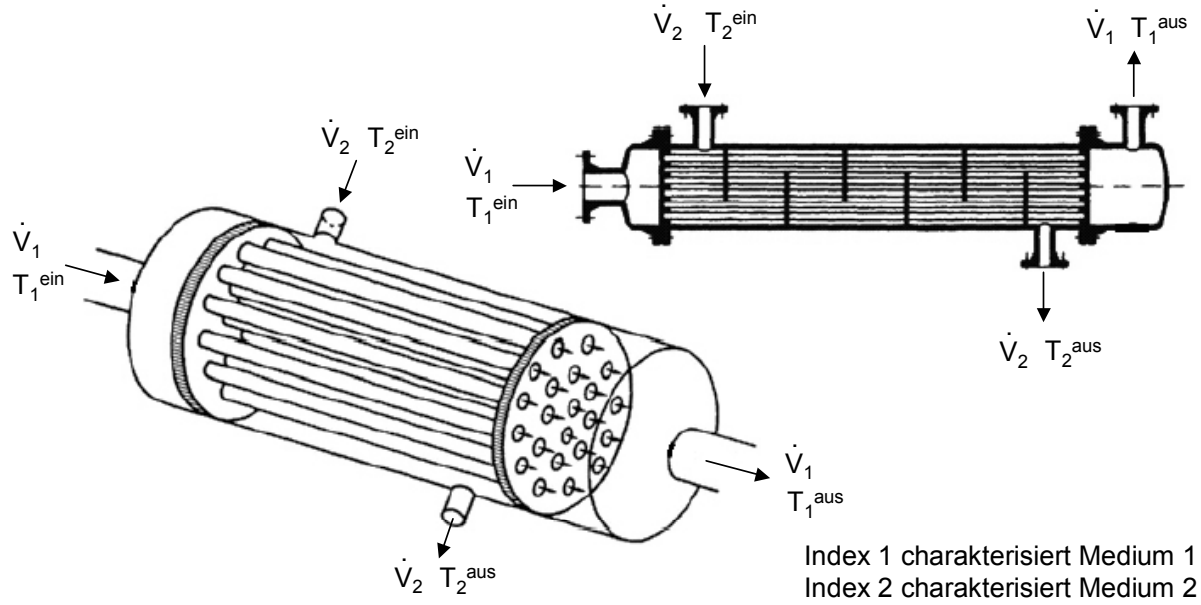


Abbildung 4-4: Rohrbündelwärmetauscher [BREHM 03]

Ein Rohrbündelwärmetauscher besteht aus geraden Rohren, die an ihren Enden in Rohrböden eingesetzt sind (s. Abbildung 4-4). Der eine Fluidstrom strömt dabei durch die Rohre, der andere umspült diese. Die Rohrbündel bestehen aus bis zu 1000 Einzelrohren. Der Zustand des Wärmetauschers soll durch das Systembedientool zu beurteilen sein. Da es sich um eine Komponente ohne real existierende Hilfsfunktionalität handelt, ist das Partialsystem durch geeignete Feldgeräte zu ergänzen.

Der Schwerpunkt der Betrachtungen liegt dabei nicht auf der Prozessführung. Hierzu sind auch gerätetechnische Ergänzungen wie Stellventile und Temperaturfühler notwendig, die dazu dienen, die Ausgangstemperatur T_1^{aus} des wärmeabgebenden Stromes zu regeln. Die somit vorhandenen Feldgeräte können aber bei der Realisierung einer Diagnosefunktionalität für das Partialsystem Verwendung finden.

Das Interesse der Zustandsüberwachung des Wärmetauschers ist auf mögliche Ablagerungen gerichtet. Dazu zählen Versotterungen und Ablagerungen insbesondere durch kalkhaltigen Kesselstein. Diese führen primär zur Funktionsbeeinträchtigung des Systems. Sekundär kann dies aber auch zu irreversiblen Beschädigungen führen. Im rechtzeitigen Wartungsfall können durch Spülung des Systems, eventuell durch Zugabe von Lösungsmitteln, die Ablagerungen beseitigt werden.

Es ist eine Messgröße zu finden, durch die eine Überwachung des Systemzustands in geeigneter Weise möglich ist. Hierzu ist gegeben:

Die Wärmemenge $Q = c\ m\ \Delta T$

mit der spezifischen Wärmekapazität c, die notwendig ist, um in einem Körper der Masse m die Temperaturdifferenz $\Delta T = T^{aus} - T^{ein}$ zu erzeugen. Die Wärmekapazität ist zwar von der Temperatur abhängig, kann aber im Rahmen der benötigten Genauigkeit bei den vorliegenden technischen Berechnungen unterhalb von 200 °C als konstant angenommen werden.

Relevant für die weiteren Betrachtungen ist der Wärmestrom

$\frac{dQ}{dt} = c\frac{dm}{dt}\Delta T$ mit der Dichte $\rho = \frac{m}{V}$ und der Annahme, dass die Dichte im betrachteten Bereich gleichfalls nicht von der Temperatur abhängig ist. Somit ergibt sich für den Wärmestrom

$$\dot{Q} = c\rho\dot{V}(T^{aus} - T^{ein}) \qquad \textit{(Gleichung 1)}$$

Weiterhin ist anzunehmen, dass der Wärmetauscher keine Verluste nach außen hat, d. h. kein zusätzlicher Wärmestrom durch das Gehäuse erfolgt. Daraus resultiert die Wärmestrombilanz:

$$\dot{Q}_1^{aus} - \dot{Q}_1^{ein} = \dot{Q}_2^{ein} - \dot{Q}_2^{aus} \qquad \textit{(Gleichung 2)}$$

Ablagerungen an den Rohrwandungen beeinflussen in starkem Maße die Wärmeübertragung des Wärmetauschers. Diese Veränderungen sind anhand des Wärmedurchgangskoeffizienten zu messen und somit ein geeignetes Maß für die zuvor geforderte Zustandsüberwachung des Systems.

Die mittlere Temperaturdifferenz ΔT_m des strömenden Mediums längs der Wärmedurchgangsfläche ergibt sich zu

$$\Delta T_m = \frac{(T_1^{ein} - T_2^{ein}) - (T_1^{aus} - T_2^{aus})}{\ln\frac{T_1^{ein} - T_2^{ein}}{T_1^{aus} - T_2^{aus}}} \qquad \textit{(Gleichung 3)}$$

Der Wärmedurchgangskoeffizient k für eine mehrschichtige Rohrwand mit n Schichten ist gegeben durch

$$\frac{1}{k} = \frac{1}{\alpha_i d_1} + \frac{1}{\alpha_a d_{n+1}} + \sum_1^n \frac{1}{2\lambda_n}\ln\frac{d_{n+1}}{d_n} \qquad \textit{(Gleichung 4)}$$

mit den Größen

α_i	Wärmeübergangskoeffizient am inneren Übergang
α_a	Wärmeübergangskoeffizient am äußeren Übergang
d_n	Durchmesser von innen nach außen
λ_n	Wärmeleitfähigkeiten der einzelnen Schichten von innen nach außen

Für den Wärmedurchgang in einem Wärmetauscher gilt die Grundgleichung

$$\dot{Q} = k\,A\,\Delta T_m \qquad \textit{(Gleichung 5)}$$

mit der Wärmedurchgangsfläche *A*.

Unter der Bedingung aus (Gleichung 2) ergibt sich, dass die (Gleichung 1) mit (Gleichung 5) gleichzusetzen ist. Hieraus folgt

$$c\rho\dot{V}_2(T_2^{\,aus} - T_2^{\,ein}) = kA\Delta T_m$$

aufgelöst nach der Konstanten *k* ergibt sich

$$k = \frac{c\rho}{A}\dot{V}_2\frac{T_2^{\,aus} - T_2^{\,ein}}{\Delta T_m}$$

Dies bedeutet, dass der Wärmedurchgangskoeffizient *k* zur Beurteilung des Zustands des Wärmetauschers in der Anlage messbar ist als Funktion von

$$k = f(\dot{V}_2, T_1^{\,ein}, T_1^{\,aus}, T_2^{\,ein}, T_2^{\,aus}) \qquad \textit{(Gleichung 6)}$$

Um abschätzen zu können, in welchem Maße Ablagerungen im Wärmetauscher Einfluss auf die Veränderungen der Wärmeübertragung haben, ist der Wärmedurchgangskoeffizient für ein System mit und ohne Ablagerungen zu vergleichen. Es sind folgende Größen gegeben:

α_i, α_a	2,0 kW/m²K	Wärmeübergangskoeffizient für strömendes Wasser
d_1	20 mm	Rohrdurchmesser innen
d_2	24 mm	Rohrdurchmesser außen
d_3	25 mm	Rohrdurchmesser mit den Ablagerungen außen
λ_1	0,35 kW/mK	Wärmeleitfähigkeit von Kupfer
λ_2	0,70 W/mK	Wärmeleitfähigkeit von Kalk

Dabei ist anzunehmen, dass sich ausschließlich auf der Rohraußenseite Ablagerungen bilden.
Der Wärmedurchgangskoeffizient k_g für das gereinigte System ergibt sich zu

$$k_g = \frac{1}{\frac{1}{\alpha_i d_1} + \frac{1}{2\lambda_1}\ln\frac{d_2}{d_1} + \frac{1}{\alpha_a d_2}}$$

$$k_g = \frac{1}{\frac{1}{2000 * 0{,}002} + \frac{1}{2 * 350}\ln\frac{24}{20} + \frac{1}{2000 * 0{,}0024}}\left[\frac{W}{mK}\right]$$

$$k_g = 2{,}2\left[\frac{W}{mK}\right]$$

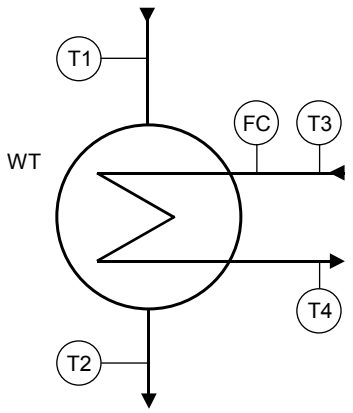
Der Wärmedurchgangskoeffizient k_k für das verkalkte System mit einer angenommen Schichtdicke des Kalks von 0,50 mm ergibt sich zu

$$k_k = \frac{1}{\frac{1}{\alpha_i d_1} + \frac{1}{2\lambda_1}\ln\frac{d_2}{d_1} + \frac{1}{2\lambda_2}\ln\frac{d_3}{d_2} + \frac{1}{\alpha_a d_2}}$$

$$k_k = \frac{1}{\frac{1}{2000 * 0{,}002} + \frac{1}{2 * 350}\ln\frac{24}{20} + \frac{1}{2 * 0{,}7}\ln\frac{25}{24} + \frac{1}{2000 * 0{,}0024}}\left[\frac{W}{mK}\right]$$

$$k_k = 2{,}0\left[\frac{W}{mK}\right]$$

Wie zu sehen ist, bewirkt eine einseitige Kalkablagerung schon eine Veränderung von 10% des ursprünglichen Wärmedurchgangskoeffizienten. Durch die direkte Proportionalität (Gleichung 5) zum Wärmestrom ist dementsprechend auch die Wirkungsweise des Wärmetauschers herabgesetzt. Dies ist in Grenzen durch die Regelung des Volumenstromes $\dot{V}_2$ kompensierbar. Allerdings wäre bei diesem Beispiel eine Wartung des Systems bei einer Überschreitung der 10%-Grenze angebracht. Um diese Veränderungen in der Anlage überwachen zu können, sind zusätzliche Messeinrichtungen notwendig. Diese ergeben sich aus (Gleichung 6) und sind der schematischen Darstellung der Abbildung 4-5 konform zu DIN 19227 und DIN 28000 zu entnehmen:



WT	Wärmetauscher
FC	Durchflussmesser für $\dot{V}_2$
T1	Temperaturmesser für T_1^{ein}
T2	Temperaturmesser für T_1^{aus}
T3	Temperaturmesser für T_2^{ein}
T4	Temperaturmesser für T_2^{aus}

Abbildung 4-5: Schematische Darstellung des Wärmetauschers

Das Partialsystem Wärmetauscher verfügt über die Prozessfunktion *Wärmetauschen*. Hinzu kommen folgende Hilfsfunktionen:

`login` `logout`	An- und Abmelden am Partialsystem durch Bedientool
`getDiagnosis`	Zustand des Wärmetauschers diagnostizieren
`setDeviceWarningParameters` `getDeviceWarningParameters`	Warnbereich für Systemzustand festlegen und auslesen
`resetDeviceWarning`	Warnungen quittieren
`getOperatingTime`	Auslesen der Betriebsstunden
`getIdentification`	Seriennummern, Versionsnummern etc. der Geräte auslesen
`checkDevice`	Gerätetest durchführen
`resetDevice`	Warmstart durchführen
`startCalibration`	Kalibrierung starten
`getCalibrationStatus`	Status der Kalibrierung auslesen
`getMaintenanceData` `setMaintenanceData`	Informationen über Wartungen auslesen und schreiben
`resetDeviceError`	Fehlermeldungen quittieren
`setDeviceErrorParameters` `getDeviceErrorParameters`	Parameter für die Auslösung von Fehlern setzen und auslesen
`restoreDeviceData` `backupDeviceData`	Sichern und Einspielen von allen Parametersätzen des Partialsystems
`setLanguage` `getLanguage`	Sprache für Benutzerdialog setzen und auslesen
`setCommunicationParameters` `getCommunicationParameters`	Kommunikationsparameter für die Anbindung an das Gesamtsystem setzen und auslesen
`...`	...

Tabelle 4-1: Hilfsfunktionen des Wärmetauschers

Wie aus Tabelle 4-1 ersichtlich, kommen zusätzlich zur Funktion `getDiagnosis` weitere Hilfsfunktionen hinzu, die zur Integration und zum Betrieb des Partialsystems unumgänglich sind. Eine Abarbeitung der Funktionen auf Partialsystemebene erfolgt verschieden durch Aufruf unterschiedlicher Funktionen der unterlagerten Geräte. Bevor auf die detaillierte Analyse der heterogenen Klassen an Hilfsfunktionen eingegangen wird, sei an dieser Stelle auf Kapitel 5 verwiesen.

Der Nutzen für den Endanwender solcher Hilfsfunktionen auf Systemebene ist unbestreitbar [LINDNER 03]. Allerdings sind einfache Wege zu finden, um den zusätzlichen Aufwand beim Engineering des Bedientools, das entsprechende

Funktionalität anbietet, zu minimieren und dem Betriebsingenieur zusätzlich die Möglichkeit einer nachträglichen Änderung und Erweiterung zu bieten.

4.2 Bedarf

4.2.1 Engineering-Konzept

Im Rahmen des DeKOS-Verbundprojektes wurde ein Vorgehensmodell zur Generierung (s. Abbildung 3-14) von Bedienobjekten für einzelne intelligente Feldgeräte entwickelt. Hierdurch kann der Gerätehersteller ohne großen Zeitaufwand einheitliche Bedienproxys in Form von DTMs generieren [BENDER ET AL. 02]. Ähnliche Ansätze sind durch Softwaretools wie das DTMstudio gegeben, durch die eine Generierung von DTMs auf Basis der Geräte-EDD möglich ist [SCHULLERER 01].

Bei den bestehenden proprietären Lösungen in der Verfahrenstechnik zur Bedienung auf Systemebene sieht die Situation etwas anders aus:

> *Den Anstrengungen der Prozessleitsystemhersteller, mit Hilfe von Asset Management Anwendungen die Verfügbarkeit und die Produktivität ihrer verfahrenstechnischen Anlagen zu erhöhen, stehen die hohen Aufwendungen des Engineerings*[48] *dieser Anwendungen gegenüber. Grund hierfür ist neben den erhöhten Konfigurationsanforderungen in Folge der gestiegenen Funktionalität der Systeme das Fehlen eines herstellerunabhängigen Gesamtmodells, woraus sich eine divergierende Vielfalt von Bedienkonzepten und -werkzeugen für Feldgeräte und Partialsystemen ergibt* [EPPLE 03].

Dementsprechend ist es notwendig, ein standardisiertes Engineering-Konzept zur einheitlichen Generierung von Bedienobjekten auf Partialsystemebene zur Verfügung zu stellen, das domänenunabhängig sowohl für die Fertigungs- als auch die Verfahrenstechnik geeignet ist. Denn nur so ist eine Reduzierung der Engineering-Kosten bei der Erstellung der Software und ein Annehmen der Lösung durch die Hersteller und Endanwender gewährleistet.

4.2.2 Modellierung und Beschreibung

Es gilt, die Komplexität der Abläufe und Systeme durch eine geeignete Modellbildung zu reduzieren. Erst dadurch ist eine Beherrschbarkeit zu erzielen

[48] Im Sinne des Entwurfs, der Implementierung, Test und Inbetriebnahme einer Software.

[AHRENS/SCHEURLEN/SPOHR 97]. Nach LAUBER und GÖHNER ist es ein Kennzeichen der wissenschaftlichen Bearbeitung einer technischen Aufgabe, dass die Welt der Realität mit Hilfe einer Welt der Modelle untersucht wird [LAUBER/GÖHNER 99A]. Die dabei anzuwendende Methode bei der Erstellung solcher Modelle ist, Modellelemente durch einen Abstraktionsvorgang unter Anwendung eines Modellierungskonzeptes zu gewinnen und durch grafische, textuelle oder mathematische Beschreibungen darzustellen.

Durch ein so gewonnenes allgemeingültiges Modell des zuvor beschriebenen Partialsystemproxys, eingebunden in einer Umgebung bestehend aus dem Benutzerdialog und der unterlagerten Geräte, ist zu untersuchen, unter welchen Voraussetzungen ein standardisiertes und komfortables Verfahren zur Generierung entsprechender Proxys realisierbar ist. Das Modell des Partialsystems ist dabei einheitlich zu beschreiben. Es ist eine eindeutige, formale und plattformunabhängige Notation zu spezifizieren. Diese ist essentieller Bestandteil des Lösungskonzeptes. Das Modell umfasst dabei u. a. die angebotene Hilfsfunktionalität des Partialsystems, die Abhängigkeiten zwischen den unterlagerten Geräten und die Verweise auf diese.

Aus der spezifizierten Referenzarchitektur des allgemeinen Partialsystemproxys können Beschreibungen abgeleitet werden, aus denen wiederum eine Software-Generierung der realen Partialsystemproxys durchführbar ist. Das Vorgehensmodell und die Tools zur Generierung von Geräteproxys auf Basis der FDT-Technologie als Ergebnis des Verbundprojektes DeKOS sind dazu prädestiniert [BENDER ET AL. 02]. Es ist eine geeignete deklarative Metasprache[49] zu wählen, durch die eine einfache Weiterverarbeitung der Dokumente softwareseitig sichergestellt ist.

Ein Hauptvertreter der deklarativen Metasprachen, der sich nicht nur im Bereich der Automatisierungstechnik jüngst etabliert hat, ist die Extensible Markup Language (XML) [BIRKHOFER 01]. Es sind im Umfeld der Automatisierungstechnik diverse Einsatzbereiche zu finden, in denen XML erfolgreich eingesetzt wird. So handelt es sich beispielsweise bei den PROFInet-Komponentenbeschreibungen aus Kapitel 3.3.2, die zur Weiterverarbeitung in ein Verschaltungstool eingelesen werden, um XML-Dokumente [PNO 01A]. SPERBER geht davon aus, dass sich XML als Standardformat durchsetzen wird, denn die Verfügbarkeit von leistungsfähigen Werkzeugen forciert die Verwendung [SPERBER 03].

[49] Sprache, die vor allem zum Zwecke der Beschreibung eingesetzt wird.

4.2.3 Klassifikation der Hilfsfunktionen

Um ein Vorgehensmodell zur Generierung der Partialsystemproxys auf Basis des zuvor beschriebenen Modells entwickeln zu können, sind die Eigenschaften des Systems und der unterlagerten Geräte exakt zu untersuchen. Erste Ansätze sind [BLUM 02] zu entnehmen.

Eine Ordnungssystematik für die standardisierte Hilfsfunktionalität des DeKOS-Bedienmodells wäre für den Gerätehersteller von Vorteil. Seitens der Partialsystemproxys wird eine gewisse Grundfunktionalität benötigt, die von den unterlagerten Geräten zu erbringen ist [BLUM 02].

Das Verhalten zwischen den Funktionen des Partialsystems und der unterlagerten Geräte ist zu untersuchen. Die Eigenschaften der jeweiligen Klassen bezüglich ihrer Abarbeitung sind zu formulieren und in die Referenzarchitektur zu integrieren. Die Analyse des Stands der Technik in Kapitel 3 hat ergeben, dass diverse Ansätze zur Ordnung der Gerätefunktionalität existieren. Diese sind bei der Klassifikation der Hilfsfunktionen zu berücksichtigen.

4.3 Defizite der DRA

Die Ausführungen in Kapitel 3 haben gezeigt, dass das DeKOS-Bedienmodell für die Kompensation der analysierten Defizite bei der Bedienung auf Partialsystemebene geeignet ist. Die bestehende Referenzarchitektur DRA zur Abbildung des spezifizierten Bedienmodells ist nachfolgend hinsichtlich ihrer Vollständigkeit und Flexibilität in Bezug auf die Beschreibung von Partialsystemen zu untersuchen. Eine Formulierung der notwendigen Maßnahmen zur Weiterentwicklung der DRA folgt.

4.3.1 Strukturierung der Hilfsfunktionen

Um die Menge der standardisierten Hilfsfunktionen leicht handhaben zu können und der Forderung aus Kapitel 4.2.3 nach einer verfügbaren Grundfunktionalität nachzukommen, ist die angesprochene Ordnungssystematik in geeigneter Weise umzusetzen. Eine vollständige Klassifikation der Hilfsfunktionen ist dem Hersteller bei der Entwicklung seiner Geräte an die Hand zu geben.

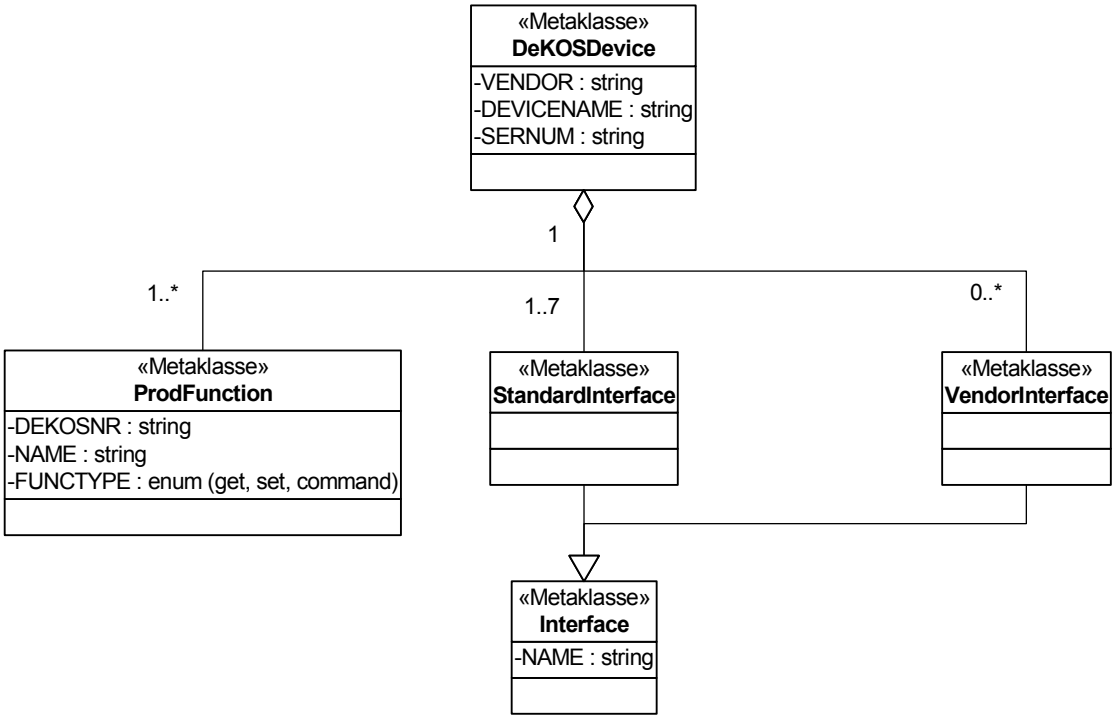


Abbildung 4-6: Klassenmodell der DRA, Gerätesicht

Nachfolgend ist zu untersuchen, ob die bestehende Architektur geeignet ist, um die geforderte Ordnungssystematik abzubilden. Hierzu wird das vereinfachte Klassenmodell[50] der DRA abschnittsweise näher betrachtet.

Das Gesamtgerät wird abgebildet in der Klasse `DeKOSDevice` (s. Abbildung 4-6). Die Attribute ermöglichen dem Modellierer die allgemeinen Geräteeigenschaften näher zu beschreiben. Die Klasse `ProdFunction` enthält die Prozessfunktion (s. Definition 2.6), d. h. die Kernfunktion des Feldgerätes. Die Klasse kann beliebig oft, muss jedoch mindestens einmal, instanziert werden.

Die gesamte Hilfsfunktionalität eines Feldgerätes ist gemäß dem Bedienmodell über Zugriffsrechte zugänglich (s. Abbildung 3-3). Unter Berücksichtigung der Richtlinie VDI 2187 wird das Rollenmodell durch die Klasse `StandardInterface` abgebildet. Der Namensraum der 7 Instanzen ist dabei durch die Spezifikation festgelegt. Zusätzlich ist dem Gerätehersteller aber die Möglichkeit gegeben, über die Klasse `VendorInterface` eigene Bedienerrollen zu realisieren. Beide beschriebenen Klassen übernehmen das Attribut `Name` von der abstrakten Oberklasse `Interface`.

[50] Diagramm, das eine Menge von Klassen, Schnittstellen und Kollaborationen sowie deren Beziehungen zeigt [BOOCH 99].

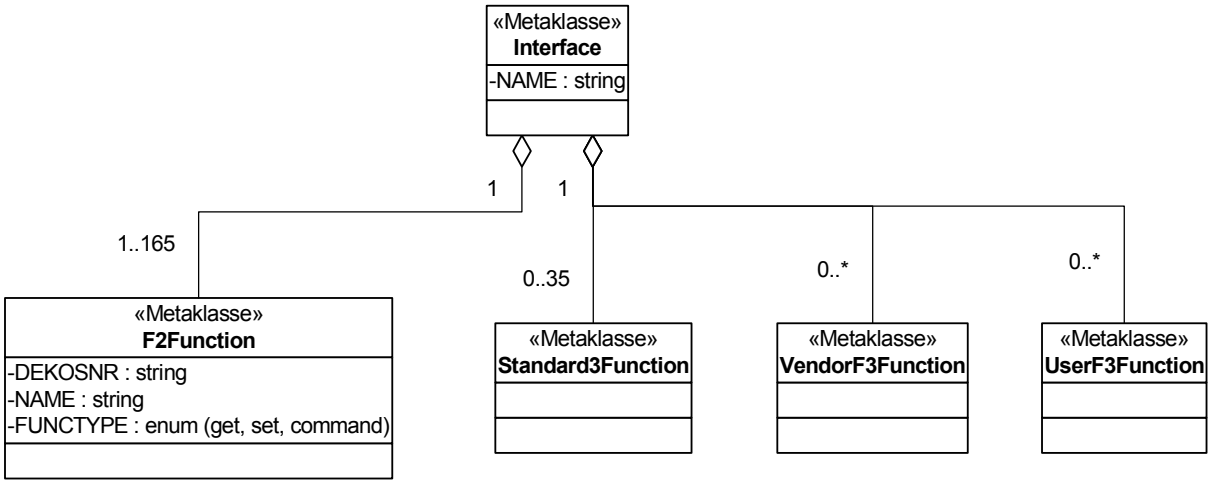


Abbildung 4-7: Klassenmodell der DRA, Hilfsfunktionen

Die Hauptaufgabe des DeKOS-Bedienmodells ist die einheitliche Abbildung der Hilfsfunktionen eines Gerätes. Diese sind wie zuvor beschrieben über die Klasse `Interface` angeordnet. Die unterschiedlichen Arten der Hilfsfunktionen (s. Kapitel 3.2):

- Atomare Funktionen
- Komfortfunktionen (vorgegeben durch Spezifikation)
- Komfortfunktionen (vom Hersteller zu erstellen)
- Komfortfunktionen (vom Anwender zu erstellen)

sind durch die instanziierbaren Klassen:

- `F2Function`
- `StandardF3Function`
- `VendorF3Function`
- `UserF3Function`

repräsentiert (s. Abbildung 4-7). Der Namensraum der 165 atomaren Funktionen und der 35 Komfortfunktionen ist äquivalent zu den Bezeichnungen der Standardinterfaces durch die Spezifikation festgelegt. Zusätzlich hat sowohl der Hersteller als auch der Endanwender die Möglichkeit beliebig viele Komfortfunktionen hinzuzufügen (s. Kapitel 3.2.2). Alle drei Komfortfunktions-Klassen übernehmen das Attribut `DEKOSNR` und `Name` von der abstrakten Oberklasse `F3Function`.

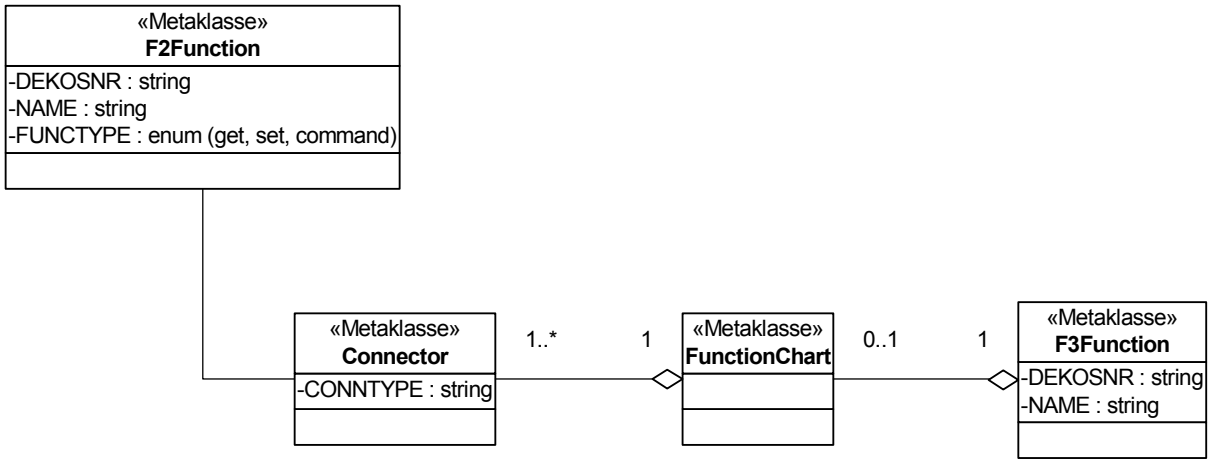


Abbildung 4-8: Klassenmodell der DRA, Abarbeitung einer Komfortfunktion

In den Instanzen der Komfortfunktionsklasse wird lediglich der Funktionsname und eine interne ID[51] in Form des Attributes `DEKOSNR` festgelegt. Die Klasse `FunctionChart` bildet (s. Abbildung 4-8) die ablauforientierten Graphen (s. Abbildung 3-2) der atomaren Funktionen in Anlehnung an die SFC[52]-Pläne [IEC 61131-3] ab. Dessen Objekte wiederum enthalten Objekte der Klasse `Connector`. Hierdurch werden die Transitionen des Funktionsablaufes repräsentiert. Zwischen den Klassen `Connector` und `F2Function` besteht eine Assoziation durch die die abzuarbeitenden atomaren Funktionen festgelegt werden.

Zusammenfassend ergibt sich, dass die analysierte Referenzarchitektur eine feste Strukturierung der Funktionen über die Interfaces vorsieht. Eine weitere Strukturierungsvorschrift wäre über zusätzliche Interfaces vom Typ `VendorInterface` denkbar. Dies ist allerdings aus nachfolgenden Gründen abzulehnen:

- Zum einen sind somit keine zusätzlichen Untergruppen realisierbar, wie dies z. B. bei Klassifikationsansätzen der FDT-Spezifikation [PNO 00], [PNO 01B] vorgesehen.
- Und zum anderen wird die Strukturierung der Hilfsfunktionen primär für den Gerätehersteller bei der Erstellung der Beschreibung benötigt. Daher ist es auch nicht sinnvoll, diese zusätzlichen Informationen in die Referenzarchitektur und somit in die Gerätebeschreibungen zu integrieren. Daraus resultiert nur eine unnötige Vergrößerung der Beschreibungen.

[51] Identifier, im System einmaliger Bezeichner zur Identifikation.
[52] Sequential Function Chart

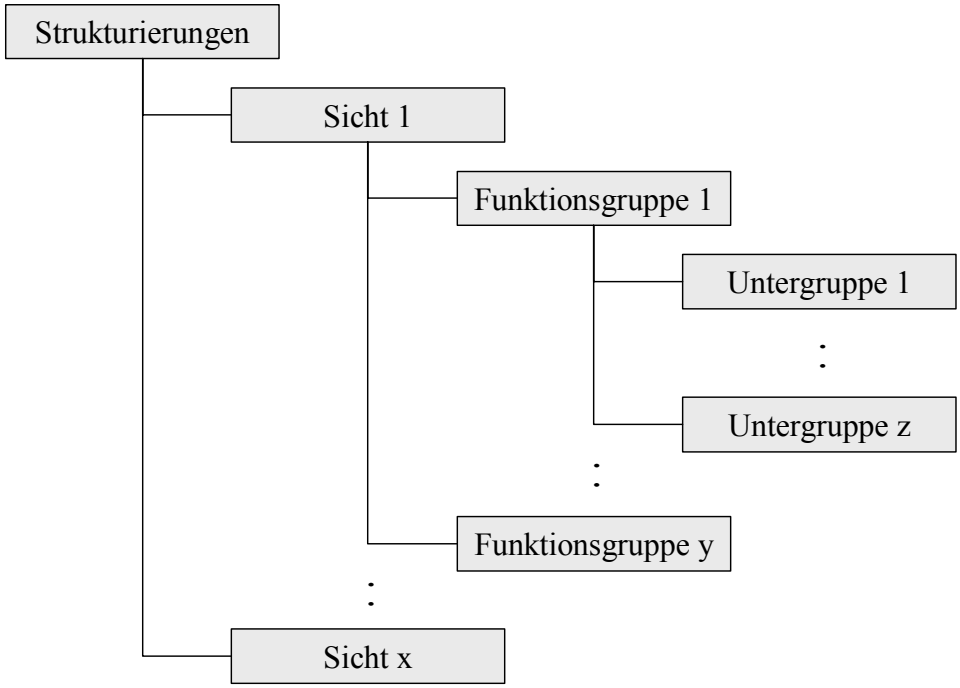


Abbildung 4-9: Schematische Darstellung des Strukturierungskonzeptes

Die angesprochene Ordnungssystematik ist in geeigneter Weise orthogonal zu der bestehenden Zuordnung der Interfaces durch ein Strukturierungskonzept (s. Abbildung 4-9) zu realisieren. Es soll lediglich eine bestimmte Sicht auf die Funktionen der Bibliothek beim Engineering darstellen und nicht Einfluss auf die Zuordnung des Rollenmodells nehmen (s. Abbildung 4-10). Inwieweit zusätzlich auch der Endanwender von einer derartigen Lösung bei der Bedienung profitieren könnte, ist zu untersuchen; denn komplexe Anlagen und Geräte verfügen auch bei einem einheitlichen Bedienmodell über eine große Anzahl an Hilfsfunktionen, die der Bediener handhaben muss [ITM 02B].

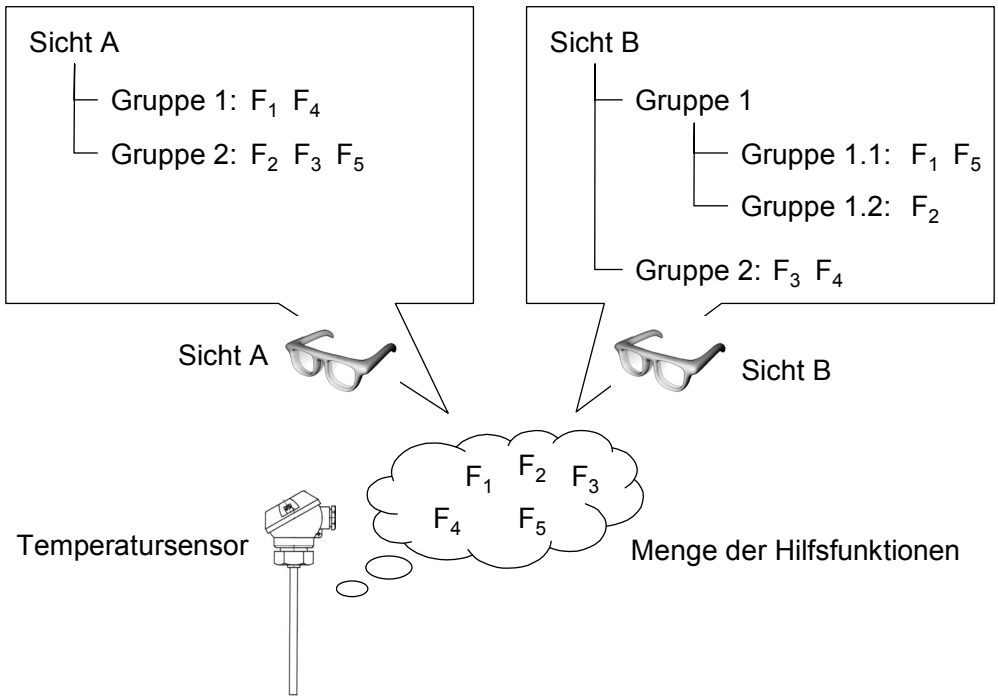


Abbildung 4-10: Unterschiedliche Sichten auf die Hilfsfunktionen eines Gerätes

Zur universellen Abbildung beliebiger Ordnungssystematiken soll die Referenzarchitektur StrucRefArchitecture spezifiziert werden. Das Klassenmodell (s. Abbildung 4-11) ist nachfolgend beschrieben:

Das Wurzelelement der StrucRefArchitecture ist die Klasse `Structures`. Neben dem Attribut `Name` verfügt jedes Objekt der beschriebenen Klasse über ein Objekt der Klasse `Fileinfo` zur detaillierten Beschreibung der Ordnungssystematik über die Attribute `AUTHOR`, `DATE`, `VERSION` und `FILETYPE`. Jedes Objekt der Klasse `Structures` verfügt über mindestens eine Strukturierungssicht der Klasse `Structure`. Zusätzlich zum Attribut `Name` ist über eine Aggregation zur Klasse `Description` eine Beschreibung in Anlehnung an die DeKOS Spezifikation möglich.

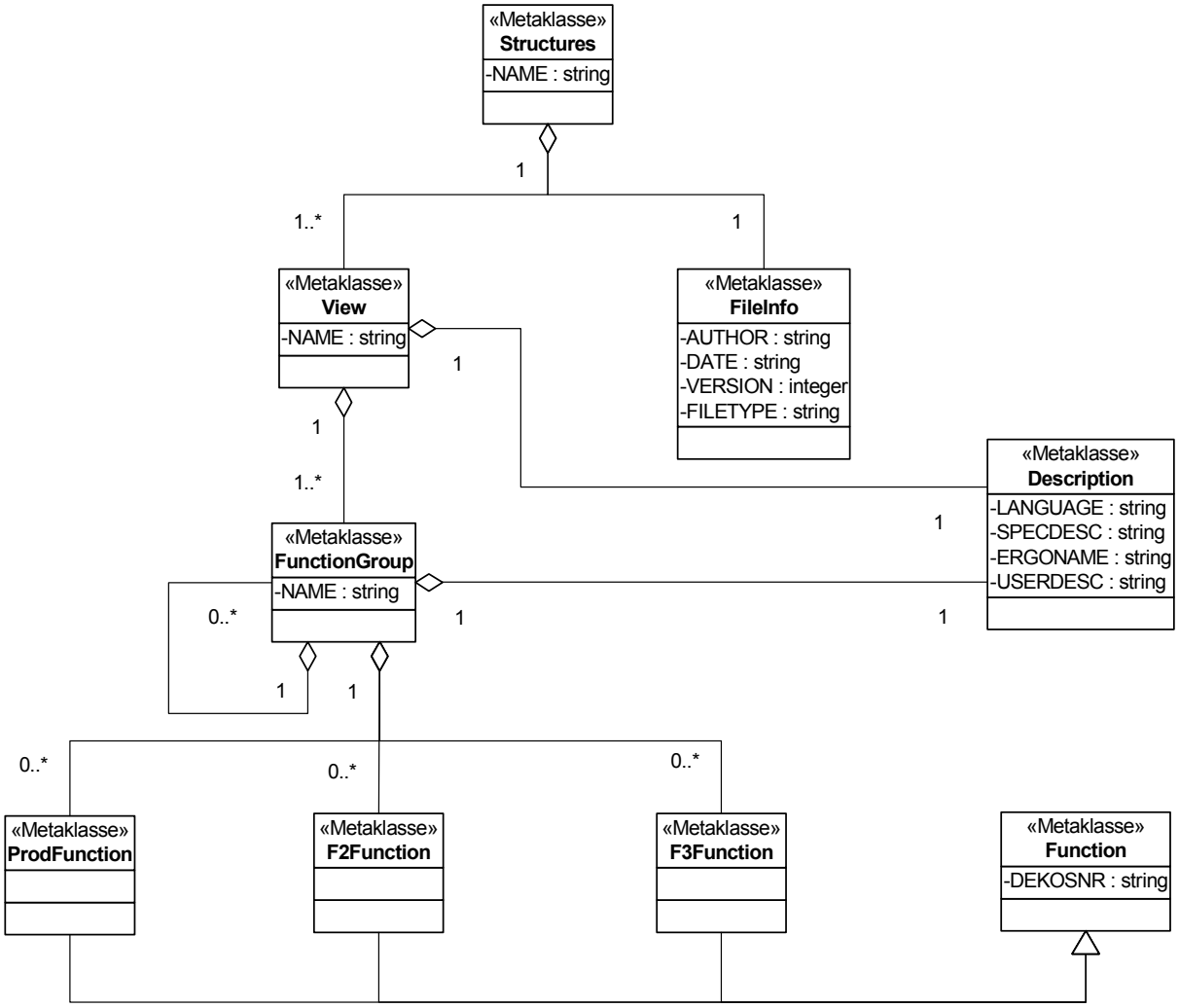


Abbildung 4-11: Klassenmodell der StrucRefArchitecture

Wie in Abbildung 4-9 schematisch dargestellt, verfügt jedes Objekt der Klasse `View` über mindestens ein Objekt der Klasse `FunctionGroup`. Jedes Objekt dieser Klasse kann wiederum beliebig viele Objekte der gleichen Klasse instanzieren. Ein Aufbauen beliebig vieler Hierarchieebenen ist dadurch gegeben. Die Klasse `FunctionGroup` hat äquivalent zur Klasse `View` eine Aggregation zur Klasse `Description`.

Zentrales Element der StrucRefArchitecture ist die Zuordnung der Funktionen zu den einzelnen Funktionsgruppen. Die Klassen `ProdFunction`, `F2Function` und `F3Function` erben von der abstrakten Oberklasse `Function` das Attribut `DEKOSNR`. Hierbei handelt es sich wie zuvor beschrieben um eine interne ID zur Referenzierung der Hilfsfunktionen.

4.3.2 Komplettierung der Funktionsbibliothek

Das DeKOS-Bedienmodell sieht eine feste Anzahl an standardisierten atomaren Funktionen und Komfortfunktionen vor (s. Kapitel 3.2). Diese werden durch die DRA hinsichtlich des Namensraumes und der Semantik definiert.

BREGULLA stellt in [BREGULLA 01] die These auf, dass Feldgeräte verschiedener Hersteller und aus beliebigen Geräteklassen mit den definierten Funktionen hinsichtlich ihrer Hilfsfunktionalität vollständig zu beschreiben sind. Es ist zu untersuchen, inwieweit die Hilfsfunktionen, die aus einer Analyse der Funktionalität einzelner intelligenter Feldgeräte durch einen kreativen Prozess abstrahiert wurden, auch zur vollständigen Beschreibung der Hilfsfunktionalität mechatronischer Systeme geeignet sind.

Dieser Sachverhalt ist durch die exemplarische funktionale Beschreibung von geeigneten mechatronischen Systemen aus dem Bereich der Verfahrens- und der Fertigungstechnik zu untersuchen. Eine eventuelle Erweiterung der Funktionsbibliothek ist anschließend in die Referenzarchitektur aufzunehmen.

4.3.3 Integration erweiterter Objekteigenschaften

Ein zu erstellendes Metamodell zur standardisierten Abbildung der Hilfsfunktionalität beliebiger mechatronischer Systeme verfolgt das Ziel die Eindeutigkeit und Implementierungsunterstützung und damit die Anwendbarkeit der vorgeschlagenen Lösungsarchitektur zu verbessern. Die Auswertung des Stands der Technik in Kapitel 2.4 ergab, dass das DeKOS Bedienmodell als Grundlage hierzu sehr gut geeignet ist.

Die Eigenschaften der Modelle zur Beschreibung der Hilfsfunktionalität von Geräten und Partialsystemen sind sehr ähnlich. Im einzelnen bedeutet dies, beide verfügen über:

- Interfaces zur Abbildung des Rollenmodells
- Atomare Funktionen
- Komfortfunktionen
- Abläufe der Komfortfunktionen

Zur Beschreibung der Hilfsfunktionalität von Partialsystemen sind zusätzlich allerdings weitere Informationen notwendig, die in einem erweiterten Modell Berücksichtigung finden sollen. In Kapitel 3.4.2 wurde beim Lösungsansatz beschrieben, wie eine Funktion auf Partialsystemebene durch Ausführen einer oder auch mehrerer Funktionen der unterlagerten Geräte ausgeführt wird. Diese zusätzlichen Informationen sind in dem erweiterten Modell mit aufzunehmen. Äquivalent zu den Abläufen der atomaren Funktionen bei der Ausführung einer

Komfortfunktion sind die Ausführungen der Partialsystemfunktionen durch die Funktionen der unterlagerten Geräte zu modellieren. Hieraus ergeben sich weitere benötigte Eigenschaften:

- Verknüpfung der Funktionen der unterlagerten Geräte an die Partialsystemfunktion
- Referenzierung der verschiedenen Geräte
- Integration einer Abarbeitungslogik (Boolesche Verknüpfungen, komplexer Algorithmus etc.)

Zur Abbildung der Hilfsfunktionalität von Partialsystemen soll die Referenzarchitektur SysRefArchitecture spezifiziert werden. Diese basiert auf dem in Kapitel 4.3.1 eingeführten Klassenmodell der DRA.

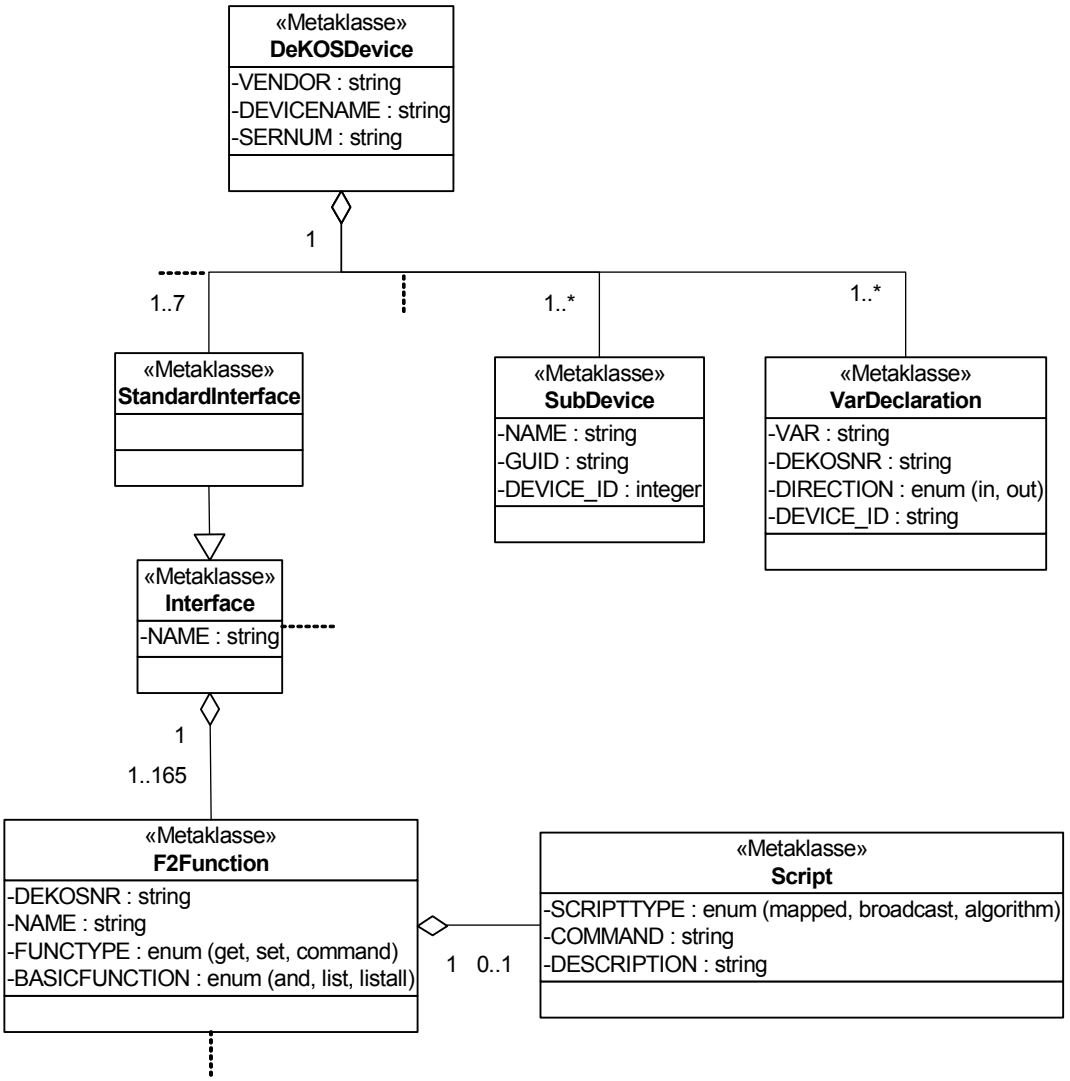


Abbildung 4-12: Ausschnitt aus dem Klassenmodell der SysRefArchitecture

Der Ausschnitt des Klassenmodells aus Abbildung 4-12 ist näher zu betrachten. Das vollständige Klassenmodell der SysRefArchitecture ist dem Anhang A zu entnehmen. Als Schnittstelle für die Erweiterung sind u. a. die atomaren Hilfsfunktionen des Partialsystems abgebildet in der Klasse `F2Function` gewählt. Ob die Hilfsfunktion ein ausführbares Skript beinhaltet, wird über das Attribut `BASICFUNCTION` festgelegt. Es handelt sich dabei um einen Aufzählungstypen mit den Werten `and`, `list` und `listall`. Die Ankopplung der unterlagerten Funktionen an die auszuführenden Partialsystemfunktionen erfolgt in Form einer 1-zu-1-Beziehung über die Klasse `Script`. Die Instanzen dieser Klasse enthalten in Anlehnung an höhere Programmiersprachen die Skripten[53] zur Abarbeitung der Funktionen in Form des Attributs `COMMAND`. Über das Attribut `SCRIPTTYPE` wird die Art der Ausführung unterschieden.

Im einfachsten Fall wird eine Funktion auf Partialsystemebene durch die Ausführung der gleichen Funktion auf nur einem bestimmten unterlagerten Feldgerät ausgeführt (`SCRIPTTYPE = mapped`). Die Funktion kann somit unverändert an das Gerät weitergeleitet werden (s. Abbildung 4-13). Eine Weiterleitung eventueller Rückgabewerte an die Ursprungsfunktion des Partialsystems erfolgt.

```
Subsystem.setCommunicationParameters()
     Communicationprocessor.setCommunicationParameters()
```

Abbildung 4-13: Beispiel eines Funktionsaufrufes (SCRIPTTYPE = mapped)

Neben diesen Funktionen mit einfacher Logik bei der Ausführung wird weiter zwischen dem `SCRIPTTYPE broadcast` und `algorithm` unterschieden.

```
Subsystem.getDeviceError()
     Device1.result = Device1.getDeviceError()
     Device2.result = Device2.getDeviceError()
     if Device1.result <> 0
     then returnvalue = device_id(Device1)
     if Device2.result <> 0
     then returnvalue = device_id(Device2)
     if Device1.result = 0 and Device2.result = 0
     then returnvalue = 0
```

Abbildung 4-14: Beispiel eines Funktionsaufrufes (SCRIPTTYPE = broadcast)

[53] Übergeordnete Sprache, um vorhandene Programme oder Prozeduren kontrolliert nacheinander oder miteinander kooperierend ablaufen zu lassen [DUDEN 01].

In Abbildung 4-14 ist ein Beispiel zur Diagnosefunktion eines Partialsystems gegeben. Die Abarbeitungsreihenfolge spielt dabei keine Rolle. Die aufgerufene Funktion wird an alle unterlagerten Geräte gleichermaßen weitergeleitet. Aus den Einzelergebnissen wird durch eine einfache Verknüpfung der Rückgabewert ermittelt. Dabei wird in dem Beispiel davon ausgegangen, dass nur ein Feldgerät gleichzeitig einen Fehler produzieren kann. Der Bediener bekommt beim Aufruf der Diagnosefunktion des Partialsystems im Fehlerfall den Namen des schadhaften Gerätes durch die interne `device_id` zurückgeliefert. Um den genauen Fehler analysieren zu können, ist anschließend die zielgenaue Abfrage des entsprechenden Gerätes durchzuführen.

Beim `SCRIPTTYPE algorithm` können komplexe Ablaufstrukturen realisiert werden. Dabei wird die Abarbeitungsreihenfolge beachtet. Gängige mathematische Funktionen, Abarbeitungsschleifen etc. stehen zur Realisierung der Hilfsfunktionen zur Verfügung.

```
Storage.setAccelerationParameters(max_a, max_v)
     Drive1.max_v = Drive1.i * 2 ^ -0.5 * max_v
     Drive1.max_a = Drive1.i * max_a
     Drive2.max_v = Drive2.i * 2 ^ -0.5 * max_v
     Drive2.max_a = Drive2.i * max_a
     Drive1.setAccelerationParameters(Drive1.max_a,
     Drive1.max_v)
     Drive2.setAccelerationParameters(Drive2.max_a,
     Drive2.max_v)
```

Abbildung 4-15: Beispiel eines Funktionsaufrufes (SCRIPTTYPE = algorithm)

Im Beispiel aus Abbildung 4-15 in Anlehnung an Kapitel 2.2.4 erfolgt eine Änderung der maximal zulässigen Geschwindigkeiten und Beschleunigungen der zwei Antriebe eines Hochregallagers. Dabei finden die Abhängigkeiten der Werte der angetriebenen Achsen durch den Funktionsaufruf Berücksichtigung. Der Bediener muss sich somit nicht mit dem Detailwissen auseinander setzten, sondern kann ohne Kenntnis der Übersetzungsverhältnisse, der unterschiedlichen Auflösungen der Inkremente etc. in den Standardeinheiten (m/s, m/s²) das Partialsystem als Ganzes abändern. Die Umrechnung auf die Inkremente erfolgt erst im Geräteproxy.

Ergänzt wird die Klasse durch das Attribut `DESCRIPTION`. Hierdurch ist eine Kurzbeschreibung der jeweiligen Ausführung der unterlagerten Funktionen möglich.

Eine Funktion der Klasse `F2function` benötigt zur Abarbeitung dieser mindestens eine Funktion eines unterlagerten Gerätes. Die Referenzierung der Gerä-

te erfolgt durch die Klasse `SubDevice`. Die Attribute ermöglichen dem Modellierer das jeweilige Gerät im System eindeutig zu identifizieren. Hierzu wird neben dem Attribut `NAME` eine eindeutige `GUID` zugeordnet, die aber erst bei der Implementierung relevant ist. Zur internen Identifizierung wird daher zusätzlich in der Planungsphase das Attribut `DEVICE_ID` vergeben.

Die Verknüpfung zwischen der aufgerufenen Funktion auf Partialsystemebene und der auszuführenden Funktion des unterlagerten Feldgerätes erfolgt im Skript. Hierzu werden interne Variablen verwendet, die in einer Instanz der Klasse `VarDeclaration` festgelegt werden. Über das Attribut `VAR` wird der Name der internen Variable des Skripts festgelegt. Die auszuführende Funktion ist durch das eindeutige Attribut `DEKOSNR` bestimmt. Des Weiteren ist die Richtung des Informationsflusses aus Sicht des Skripts durch das Attribut `DIRECTION` vom Typ `enum (in, out)` festzulegen. Die Auswahl des Gerätes erfolgt über das Attribut `DEVICE_ID`.

4.4 Zusammenfassung

Zu Beginn des Kapitels erfolgte eine Analyse der mechatronischen Systeme. Dabei war der Schwerpunkt auf den Lifecycle der Systeme bezüglich der Hilfsfunktionalität gerichtet. Im Rahmen einer Usecase[54]-Analyse ist der Nutzen einer Hilfsfunktionalität auf Systemebene in Abhängigkeit der unterschiedlichen Personengruppen betrachtet worden. Um die Vorteile eines Systembedientools und der damit verbundenen Hilfsfunktionen herauszustellen, erfolgte das ausführliche Beispiel eines Wärmetauschers aus der Verfahrenstechnik. Dabei wurde gezeigt, dass der Nutzen für den Endanwender sehr hoch ist, allerdings der notwendige Mehraufwand seitens des Engineerings im Verhältnis dazu stehen muss.

Dies ist mit einer durchgängigen Lösung von der Entstehung eines Partialsystems bis zum Dauerbetrieb möglich. Die Lösung ist primär nur für den Endanwender von Bedeutung. Da aber gerade hierbei das perfekte Zusammenspiel der einzelnen Personenkreise wie Gerätehersteller und Anlagenbauer unterstützt und beachtet werden muss, ist eine durchgängige Lösung unbedingt notwendig. Nach der Belegung des Bedarfs nach einem solchen Engineering-Konzept, der damit verbundenen Modellierung und Beschreibung wurde die Notwendigkeit der Klassifikation der standardisierten Hilfsfunktionen dargelegt.

Kapitel 3 hat gezeigt, dass die Referenzarchitektur DRA als Basis zur Modellierung der Partialsysteme sehr gut geeignet ist. Um die notwendigen Erweiterun-

[54] Anwendungsfall

gen spezifizieren zu können, fand eine genaue Untersuchung der Defizite der DRA statt. Dabei erfolgte die Belegung des Bedarfs nach Strukturierungssichten und die Spezifikation des notwendigen Klassenmodells der Referenzarchitektur StrucRefArchitecture. Abschließend erfolgte die Spezifikation der Referenzarchitektur SysRefArchitecture zur standardisierten Modellierung der Partialsystemfunktionalität, inklusive der Verknüpfungen zu den unterlagerten Geräten. Dabei sind unterschiedliche Typen von Funktionsaufrufen definiert und Beispiele dazu gegeben worden.

Durch das Klassenmodell SysRefArchitecture ist eine einheitliche Modellierungsvorschrift für Partialsysteme vorhanden. Somit sind einheitliche Lösungen möglich. Der Implementierungsaufwand ist dabei reduziert, da auf Basis der DDD des Partialsystems eine Generierung des Proxy gegeben ist.

Es gilt nun zu überprüfen, inwieweit die Erstellung einheitlicher Partialsystembeschreibungen komfortabel erfolgen kann, d. h. die Zusammenhänge der Funktionen auf Partialsystem- und Geräteebene sind zu untersuchen. Darüber hinaus ist eine vollständige Klassifikation der Hilfsfunktionen durchzuführen. Abschließend erfolgt die formale Beschreibung der zuvor spezifizierten Modelle der Referenzarchitekturen StrucRefArchitecture und SysRefArchitecture.

5 Klassifikation und formale Beschreibung

Wie mehrfach beschrieben ist das Ziel der vorliegenden Arbeit, den gesamten Engineering-Prozess zur Realisierung eines Bedienobjektes auf Partialsystemebene zu unterstützen. An diversen Stellen der vorhergehenden Kapitel wurde schon angedeutet, dass es nicht nur um die Schaffung eines Standards zur einheitlichen Beschreibung der Hilfsfunktionalität geht, sondern dass auch die Zusammenhänge der Funktionen auf Geräte- und Partialsystemebene zu untersuchen sind. Auf deren Basis ist ein Verfahren zur halbautomatischen[55] Generierung der Systembeschreibung durch die Gerätebeschreibungen denkbar. Hierdurch ist eine nicht unerhebliche Reduzierung des Zeitaufwandes bei der Erstellung gegeben.

Darüber hinaus ist eine Ordnungssystematik zur geeigneten Klassifikation der standardisierten Hilfsfunktionen zu entwickeln, bestehende Ansätze (s. Kapitel 3) sind zu analysieren und in die Lösungsfindung mit einzubeziehen. Eine detaillierte Beschreibung der daraus resultierenden Funktionsklassen und derer spezifischen Eigenschaften folgt.

Das in Kapitel 4 erarbeitete Modellierungskonzept und die hieraus entstandene Lösungsarchitektur kann nur Verwendung finden, wenn eine formale Beschreibung der Lösungskomponenten erfolgt. Hierbei handelt es sich um die zuvor spezifizierten Klassenmodelle zur Abbildung der Funktionsstrukturierungen und Systembeschreibungen. Als geeignete Form zur Beschreibung erwies sich die Extensible Markup Language (XML). Als wichtigster Vertreter der deklarativen Metasprachen erfolgt zuvor eine kurze Einführung in diese.

5.1 Grundlagen zur Ordnungssystematik

Die Klassifikation der Funktionsbibliothek hat primär das Ziel, den Geräteherstellern bei der Beschreibung seiner Feldgeräte in geeigneter Weise zu unterstützen. Dabei ist besonders wichtig, dass die Geräte eines Partialsystems einer Anlage über geeignete einheitliche Funktionen verfügen, sodass ein möglichst großer Teil der Hilfsfunktionen des Partialsystems mit minimalem Implementierungsaufwand durch das Ausführen der gleichnamigen unterlagerten Funktionen erfolgen kann. Diese Klasse der so bezeichneten Grundfunktionen ist zu spezifizieren.

---

[55] Assistentengestützte Generierung der Beschreibung

Untersuchungen im Rahmen des Verbundprojektes DeKOS haben gezeigt, dass eine Akzeptanz des spezifizierten Bedienmodells sowohl beim Gerätehersteller wie auch beim Anwender u. a. nur zu erzielen ist, wenn die standardisierten Hilfsfunktionen dem Bediener strukturiert angeboten werden. Das in Kapitel 4.3.1 spezifizierte Klassenmodell ist derart gestaltet, dass unterschiedliche Sichtweisen auf denselben Pool an Funktionen durch einfaches Umschalten der Ansicht im Softwaretool, wie in Kapitel 6 noch gezeigt wird, realisierbar sind.

Unter dem Begriff Ordnungssystematik wird hier die Definition eines Schemas, mit dessen Hilfe man Elemente nach bestimmten Kriterien sortieren und darstellen kann, verstanden. Bei der Erstellung einer Ordnungssystematik ist in erster Linie die Wahl der Ordnungskriterien entscheidend. Es existieren diverse Ansätze, wie beispielsweise Geräteprofil-Lösungen, nach denen eine Ordnung nach der Prozessfunktion erfolgt [PNO 99]. Im vorliegenden Fall ist das Augenmerk ausschließlich auf die Hilfsfunktionen gerichtet.

5.1.1 Analyse bestehender Ansätze

Die Bewertung aus Kapitel 3.4.1 ergab, dass die Norm DIN 31051, die Empfehlung NE 57 und die FDT-Spezifikation bei dem Klassifikationsansatz zu berücksichtigen sind. Diese enthalten zum Teil domänenbezogene Ansätze, die zur vorher angesprochenen Akzeptanzsteigerung bei der Lösungsfindung heranzuziehen sind.

DIN 31051

Die Norm behandelt lediglich Maßnahmen zur Instandhaltung. Dabei wird unterteilt in:

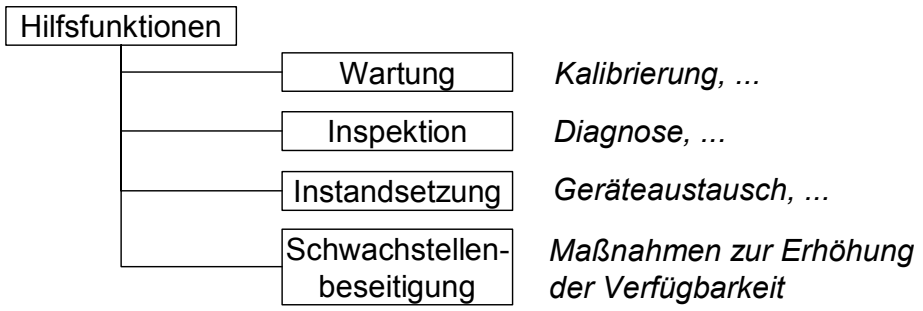


Abbildung 5-1: Strukturierung der Hilfsfunktionen nach [DIN 31051]

Der Ansatz ist sowohl für fertigungstechnische als auch verfahrenstechnische Anlagen gedacht.

NE 57

Die NAMUR-Empfehlung berücksichtigt die Anforderungen bei der Realisierung des Bedienobjektes statischer Frequenzumrichter. Die erforderlichen Hilfsfunktionen unterliegen dabei folgender hierarchischer Struktur:

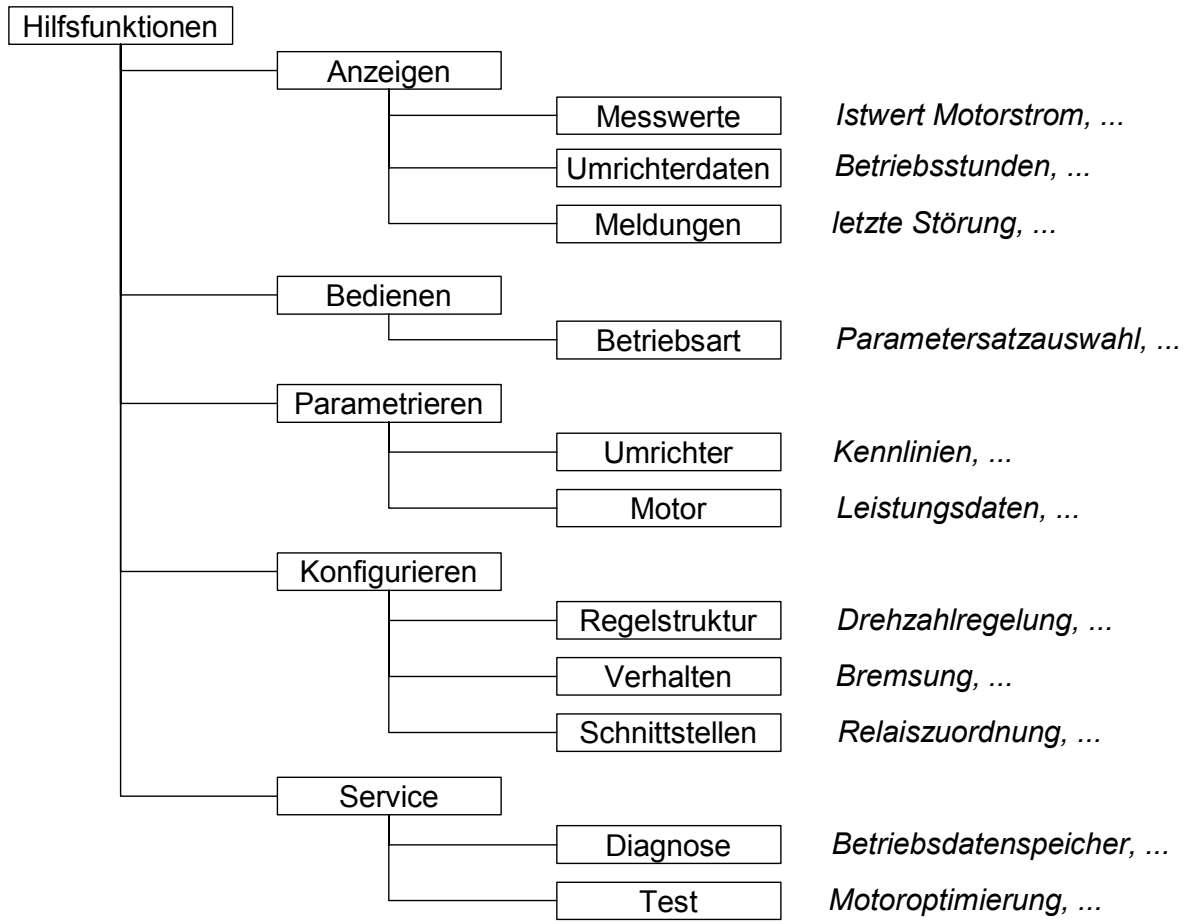


Abbildung 5-2: Strukturierung der Hilfsfunktionen nach [NE 57]

Die Empfehlung ist zwar auf Frequenzumrichter zugeschnitten, nachdem diese aber in vielen Anlagen wieder zu finden sind und eine nicht unerhebliche Komplexität aufweisen, ist der Ansatz in die Ordnungssystematik der standardisierten Hilfsfunktionen möglichst aufzunehmen.

FDT

In der Spezifikation wird neben der Hilfsfunktionalität der Feldgeräte auch die Funktionalität im Zusammenhang mit der Software (DTM, Frameapplication) strukturiert; dabei wird in Abhängigkeit der jeweiligen Benutzergruppen zugeordnet. Die Abbildung der Zugriffsberechtigungen auf die Funktionen ist in Anlehnung an das Rollenmodell [VDI 2187] (s. Kapitel 3.1.2) spezifiziert. Die Strukturierung der angebotenen Funktionen stellt sich folgendermaßen dar:

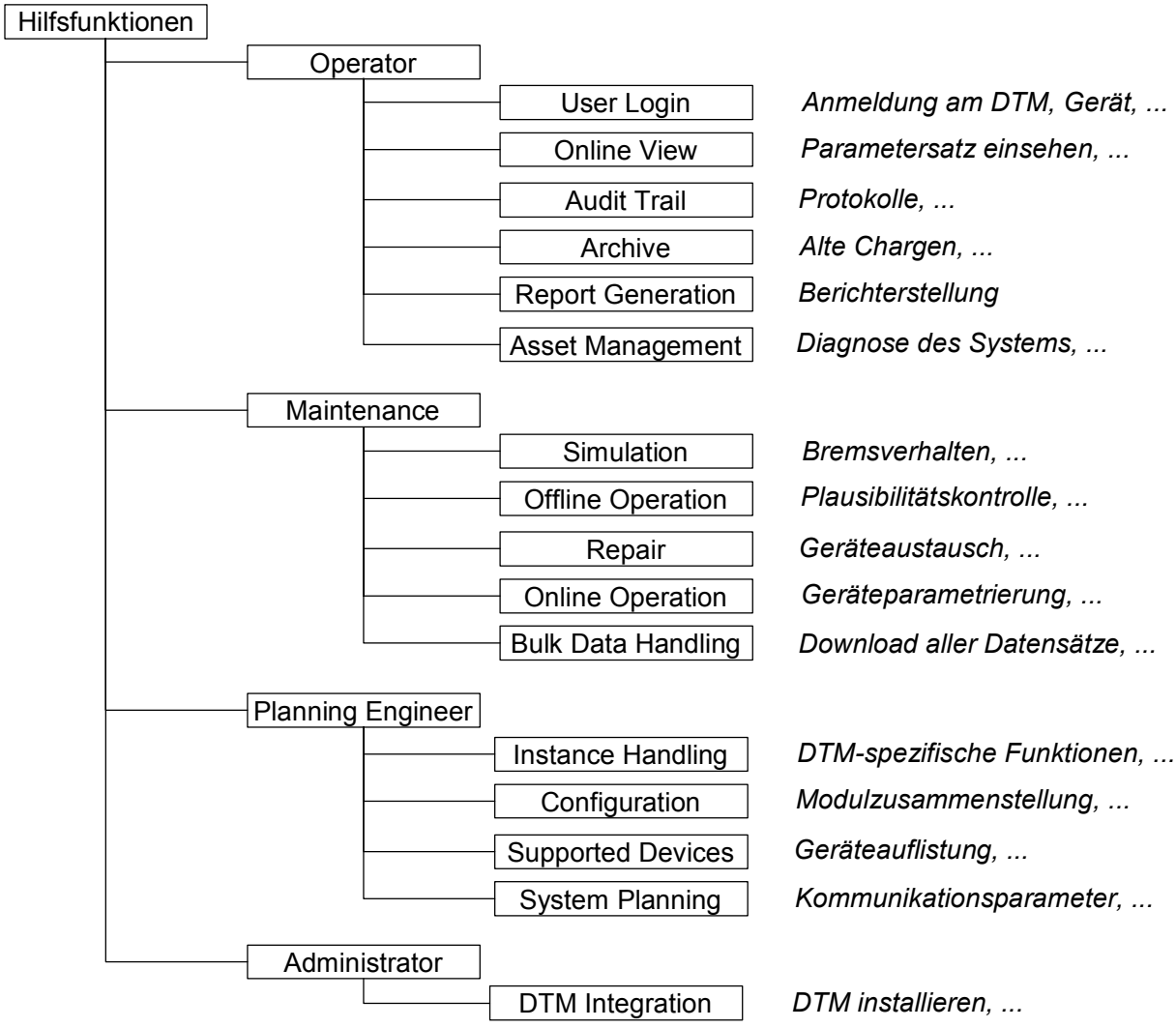


Abbildung 5-3: Strukturierung der Funktionen bei FDT nach [PNO 01B]

Die Ansätze zur Strukturierung der angebotenen Funktionen orientieren sich durchwegs an den auszuführenden Aufgaben des Personals. Hinzu kommt bei FDT das abgebildete Rollenmodell, das wiederum eine Unterteilung der Aufgabenfelder in die jeweiligen Benutzergruppen vorsieht.

5.1.2 Analyse der Abarbeitungslogik

Als Logik bezeichnet man allgemein die Wissenschaft von den Gesetzen und Formen des Denkens. Die Logik kann als eine Ethik[56] des Denkens bezeichnet werden. In der formalen Logik werden die Gesetze des abgeleiteten Wissens studiert, das aus früher bestimmten und geprüften Wahrheiten gewonnen wird, ohne in jedem konkreten Fall direkt auf die Erfahrung zurückzugreifen [SCHÖNING 00].

Im vorliegenden Fall soll die Abarbeitungslogik der Hilfsfunktionen auf Systemebene untersucht werden. Dabei sind die unterschiedlichen Ausprägungen zu analysieren, sodass eine spätere Zuordnung aller Funktionen der Bibliothek über die spezifizierte Klassifikation erfolgen kann. Untersuchungen hierzu sind [MEYER 03] zu entnehmen. Dabei wird unter Logik die Regel der Abarbeitung bezüglich der unterlagerten Funktionen verstanden. Es sei darauf hingewiesen, dass an dieser Stelle die Eindeutigkeit einer Zuordnung von Funktionen zu einer definierten Klasse noch nicht vorliegt. Daher sind die angegebenen Beispiele rein exemplarisch zu sehen. Es ist daraus noch *keine allgemeine Gültigkeit* abzuleiten.

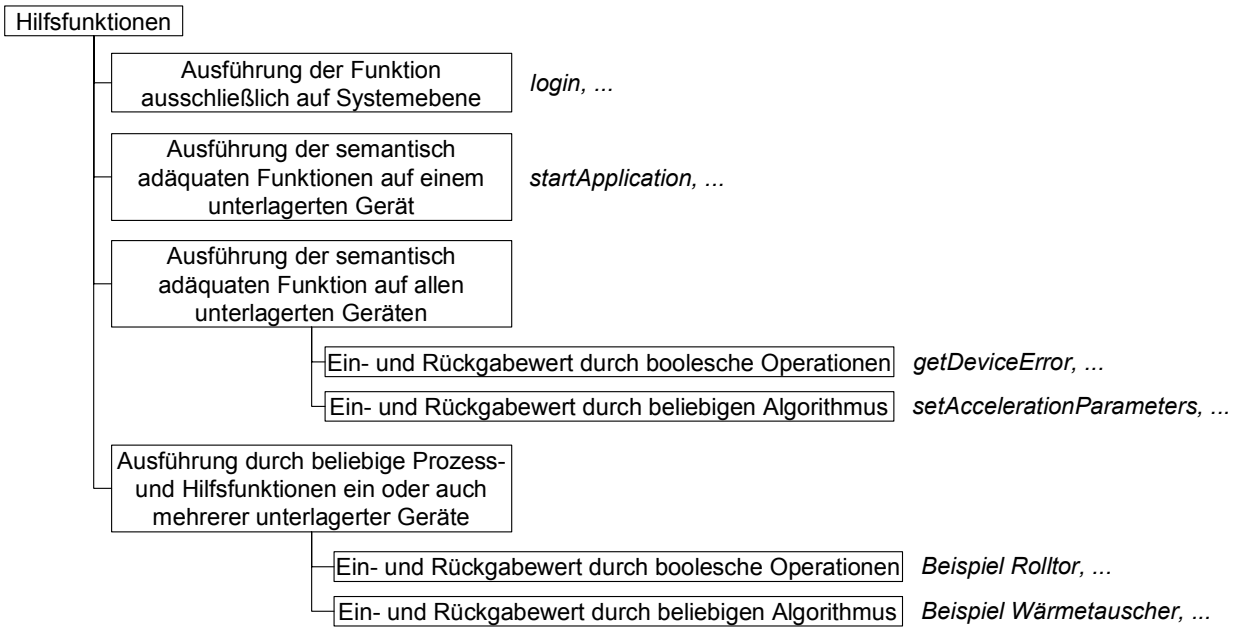


Abbildung 5-4: Strukturierung der Funktionen nach der Abarbeitungslogik

[56] Im Sinne von Handlungsregeln.

Die Strukturierung der Hilfsfunktionen in Abbildung 5-4 ist nachfolgend detailliert beschrieben. Im einfachsten Fall wird eine aufgerufene Funktion des Partialsystems nur in dessen Bedienobjekt abgearbeitet. Die unterlagerten Geräte bleiben hierdurch unberührt. Eine weitere Klasse von Funktionen bilden diese, die durch den Aufruf der semantisch adäquaten Funktion eines unterlagerten Gerätes ausgeführt werden. Die Klasse an Hilfsfunktionen die durch Ausführung der semantisch adäquaten Funktion auf allen unterlagerten Geräten abgearbeitet werden, ist weiter zu unterteilen. Die Unterklassen bilden sich aus Funktionen bei der die Wertermittlung der aufrufenden Funktion durch eine boolesche Operation, „*und*“ bzw. „*oder*“, zur Verknüpfung der Ein- und Rückgabewerte der auszuführenden Funktionen erfolgt und äquivalent hierzu durch die Wertermittlung durch einen beliebigen Algorithmus.

Die zuletzt aufgeführte Klasse ist die der Funktionen, die durch Ausführung beliebiger Prozess- und Hilfsfunktionen eines Gerätes oder auch mehrerer unterlagerter Geräte erfolgt. Diese Funktionsklasse ist u. a. essentieller Bestandteil des Asset Managements (s. Kapitel 3.3.3) und ist nur realisierbar durch ausreichendes Systemwissen. Auch bei dieser Klasse kann wieder unterteilt werden in Funktionen mit einer Verknüpfung durch eine boolesche Operation und einen beliebigen Algorithmus. Bei letzterem sei verwiesen auf das Beispiel Wärmetauscher aus Kapitel 4.1.3, bei dem die Hilfsfunktion zur Zustandsüberwachung der Komponente durch Ausführung unterlagerter Prozessfunktionen erfolgt. Nachfolgendes Beispiel aus [GROSSMANN 03] zur Verknüpfung der Werte durch eine boolesche Operation ist hierzu näher zu betrachten:

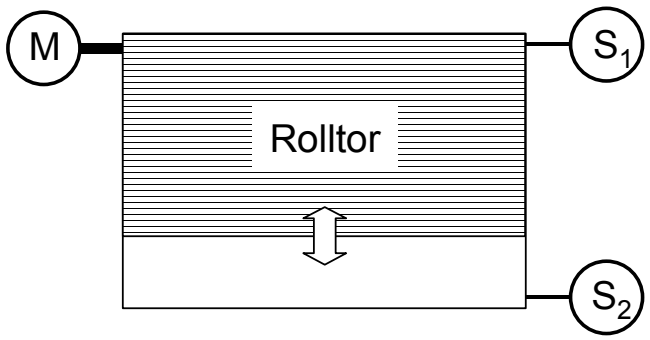


Abbildung 5-5: Beispiel zur Systemdiagnose eines Rolltores

Ein Rolltor in einer Industrieanlage sei betätigt durch den Antrieb *M*. Zur Überprüfung der Endlagen befinden sich an dem Rolltor die zwei Endlagenschalter S_1 und S_2. Die Endlagenschalter dienen primär dazu, den Antrieb zu überwachen. Ist beim Schließen „M = true“ *und* nach einer angemessenen Wartezeit „S_2.= false“ liegt ein Defekt des Antriebs *oder* ein mechanisches Problem (z. B. Verklemmung) des Gitters vor. In diesem einfachen Beispiel besagt die Logik des Systems, dass zusätzlich ein Systemfehler vorliegen muss, wenn das Rolltor

„S_1 = true“ *und* „S_2.= true“ meldet. In diesem Fall kann dann auf einen defekten Endlagenschalter rückgeschlossen werden.

5.1.3 Klassifikationsverfahren

Es existieren diverse Methoden in der Mathematik und Informatik, wie beispielsweise die Clusteranalyse, die Diskriminanzanalyse, das Machine Learning, die sich mit Klassifikationsverfahren beschäftigen. Allgemeines Ziel ist dabei die Ableitung einer Zuordnungs- oder Allokationsregel. Es ist zwischen a posteriori[57] und a priori[58] Verfahren zu unterscheiden. Letztere dienen dazu, Personen, Objekte etc. aufgrund ihrer Merkmale und Eigenschaften einer von mehreren zuvor festgelegten Klassen, Populationen oder Kategorien zuzuordnen oder die für solch eine Zuordnung wichtigsten Merkmale zu finden [NOTHNAGEL 99]. Die Zuordnungsregel wird von einer Stichprobe schon klassifizierter Personen, Objekte etc. abgeleitet. Dieser allgemeine Zugang ermöglicht die Anwendung von so genannten Diskriminanzanalyseverfahren in den verschiedensten Bereichen praktischer Problemstellungen. Einsatzbereiche hierzu finden sich in der Medizin (Diagnostik), Biologie (Auszählung von Kolonien) oder auch im industriellen Umfeld (Pattern Recognition[59]). Mit speziellen Verfahren kann beispielsweise untersucht werden, mit welcher Trefferwahrscheinlichkeit eine Zuordnung zu einer Klasse stattgefunden hat.

Da im vorliegenden Fall eine Klassifizierung von bekannten Objekten mit begrenzter Anzahl erfolgen soll, ist eine a posteriori Betrachtung wie die Clusteranalyse geeignet. Das Ziel dieser ist die Zusammenfassung der zu klassifizierenden Objekte zu Klassen, sodass die Objekte innerhalb einer Klasse möglichst ähnlich und die Klassen untereinander möglichst unähnlich sind [BACKHAUS ET AL. 94].

Eine Klassifizierung von M bestehend aus m durchnummerierten Hilfsfunktionen $(a_1, ..., a_m)$ besteht darin, n Teilmengen $\{a_x, a_y, ...\}$ *mit* $x \neq y$ (auch Klassen, Kategorien, Cluster genannt) $C_j \subset M$ zu identifizieren, die diese m Objekte beinhaltet und den folgenden Anforderungen gerecht werden:

$$C_1 \cup C_2 \cup ... \cup C_n = M$$
$$C_i \cap C_k = \emptyset \quad (i \neq k)$$
$$m_j := \left|C_j\right| \geq 1 \quad (j = 1,2,...,n) \quad \text{wobei gilt}: 1 \leq n \leq m$$

[57] Im Nachhinein durchgeführte Untersuchung.
[58] Im Vorfeld durchgeführte Untersuchung.
[59] Mustererkennung

Der erste Schritt zur Klassifizierung nach der Clusteranalyse ist die Auswahl eines geeigneten Verfahrens zur Klassenbildung. Hierzu ist zunächst die Ähnlichkeit zwischen den zu klassifizierenden Funktionen zu ermitteln. Die Homogenität bzw. die Heterogenität kann durch einen paarweisen Vergleich der zu klassifizierenden Hilfsfunktionen mittels einer statistischen Maßzahl (Distanzmaß) quantifiziert werden [BACHER 96]. Das Verfahren führt im vorliegenden Fall zum Ziel, da es nur auf endliche Mengen von Funktionen angewendet wird.

In der Literatur sind zahlreiche Verfahren zur Clusteranalyse, wie graphentheoretische oder hierarchische Verfahren, beschrieben [NUNGE 01]. Informationen über die zu klassifizierenden Objekte sind verfügbar in der Form der Analyse aus Kapitel 5.1.1 und Kapitel 5.1.2 und sind hierzu heranzuziehen. Als geeignetes Verfahren soll die hierarchisch agglomerative Clusterbildung nach BACHER Anwendung finden. Der Algorithmus ist durch folgende Schritte gekennzeichnet:

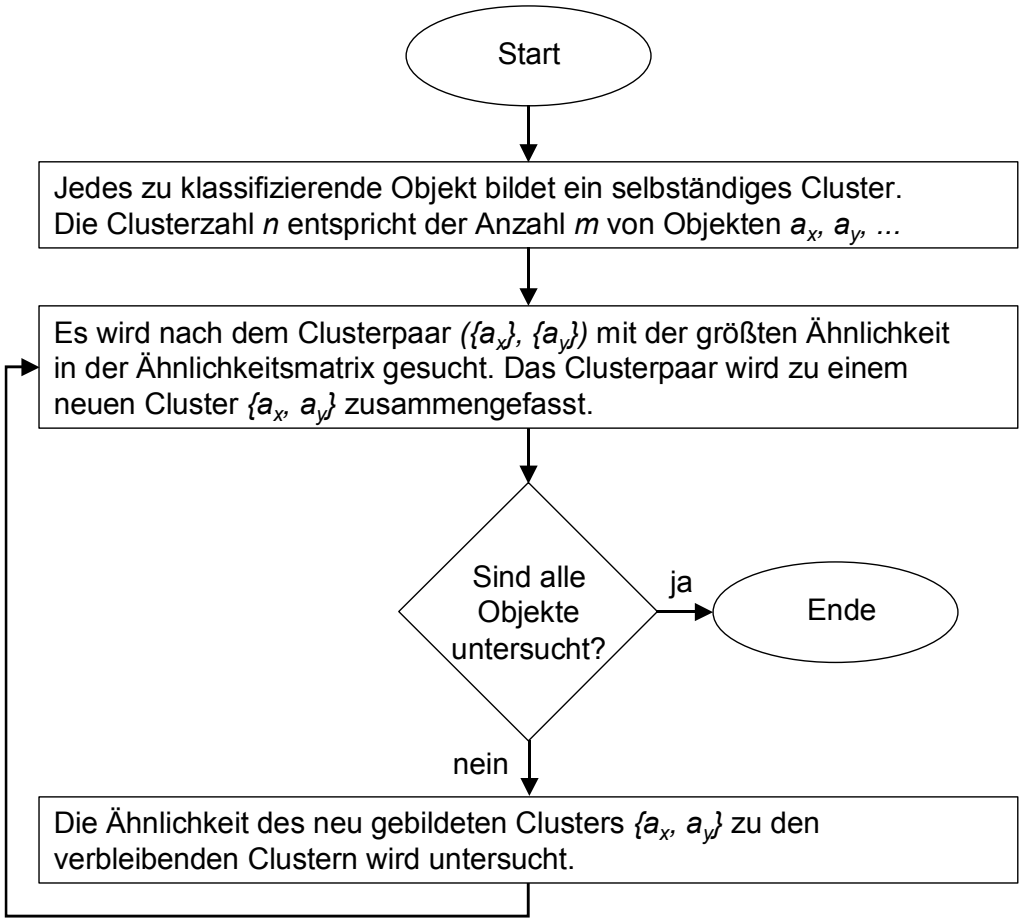


Abbildung 5-6: Algorithmus zur hierarchisch agglomerativen Clusterbildung

Die praktische Durchführung einer Clusteranalyse kann z. B. unter zu Hilfenahme des statistischen Computerprogramms SPSS[60]-Categories von der Firma SPSS Inc. erfolgen.

5.2 Funktionsklassen und deren Eigenschaften

Die nachfolgende Klassifikation beruht auf den Analyseergebnissen aus Kapitel 5.1. Die Verifikation des Klassifikationsansatzes und die Bildung von Unterklassen wurden durch Clusteranalyseverfahren (s. Kapitel 5.1.3) durchgeführt.

5.2.1 Strukturierungssichten

Bei näherer Betrachtung der Vorgehensweise zur Realisierung eines Systembedientools hat sich herauskristallisiert, dass unterschiedliche Sichtweisen (s. Kapitel 4.3.1) auf die einheitlichen Hilfsfunktionen der Bibliothek in Abhängigkeit des Arbeitsschrittes von Vorteil sind. Das Verfahren zur Generierung eines Bedienobjektes für ein Partialsystem einer Anlage stellt sich nach Abbildung 5-7 folgendermaßen dar:

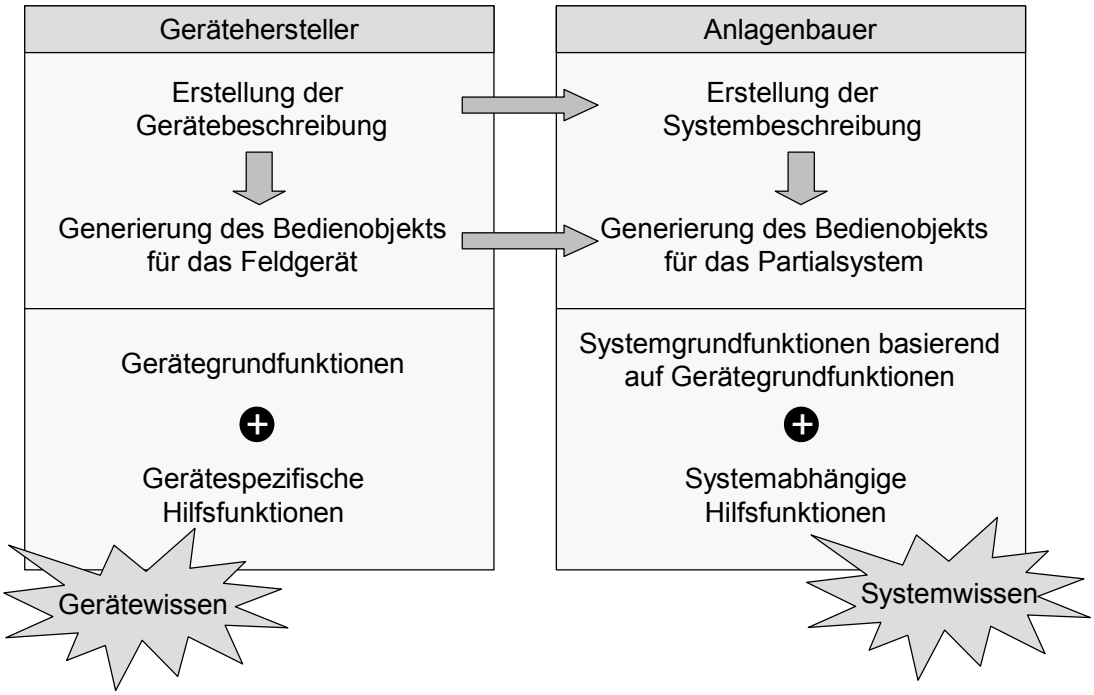


Abbildung 5-7: Generierung eines Bedienobjekts für ein Partialsystem

[60] Superior Performing Software System

Der Geräteentwickler erstellt die standardisierte Beschreibung des Feldgerätes. Dabei setzt sich die Menge der Hilfsfunktionen eines Gerätes zusammen aus Gerätegrundfunktionen, über die jedes Feldgerät verfügt und gerätespezifische Hilfsfunktionen, die der Hersteller aufgrund seines Gerätewissens aus der Menge der vorgegebenen Funktionen der Bibliothek auswählt.

Somit ergeben sich aus dem Blickwinkel des Geräteherstellers zwei unterschiedliche Sichtweisen auf die Funktionsbibliothek (s. Abbildung 5-8). Die erste repräsentiert die Gerätegrundfunktionen unabhängig von einer bestimmten Ordnungssystematik, die zweite stellt dem Entwickler alle Funktionen klassifiziert zur Verfügung, sodass dieser die notwendigen Hilfsfunktionen schnell und einfach auswählen kann. Auf Basis der vollständigen Gerätebeschreibung kann anschließend das Bedienobjekt assistentengestützt generiert werden [BENDER ET AL.02].

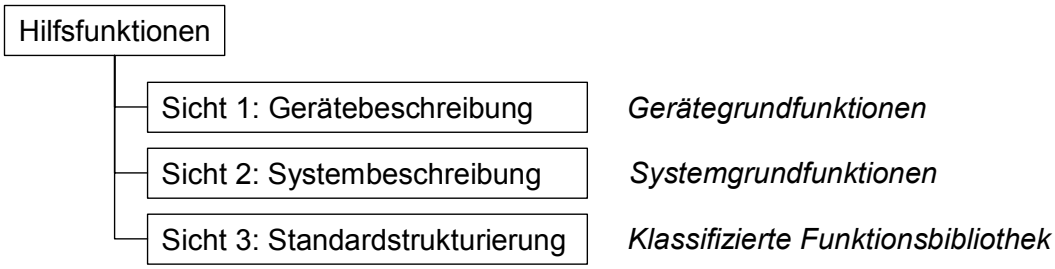


Abbildung 5-8: Definierte Strukturierungssichten

Bei der Erstellung der Partialsystembeschreibung müssen die vollständigen Beschreibungen der unterlagerten Feldgeräte zuvor erstellt werden oder schon vorhanden sein. Äquivalent zu den Gerätegrundfunktionen werden in einer Sicht die Systemgrundfunktionen ohne Ordnungssystematik aufgeführt (s. Abbildung 5-8). Darüber hinaus hat der Entwickler die Möglichkeit, die Beschreibung auf Basis seines Systemwissens mit geeigneten Hilfsfunktionen zu ergänzen. Hierzu kann auch dieser auf die Sicht mit der klassifizierten Funktionsbibliothek zugreifen. So ist in diesem Arbeitsschritt ein komfortables Auffinden der geeigneten Hilfsfunktion zur Erstellung der Systembeschreibung gewährleistet. Ein beliebiges Wechseln zwischen den verschiedenen Strukturierungssichten ist jederzeit möglich, weil ausschließlich eine Menge von zuvor standardisierten Funktionen betrachtet wird (s. Kapitel 3.2.2). Die Generierung des Bedienobjektes für das Partialsystem gestaltet sich in gleicher komfortabler Weise wie für das einzelne Feldgerät. Der Hauptunterschied liegt in der realen Ausführung einer Funktion geräteseitig, die im Proxy abgearbeitet wird. Es sei an dieser Stelle auf die Ausführungen in Kapitel 6 zur Softwarearchitektur des Bedienproxys verwiesen.

Die einheitlichen Hilfsfunktionen der standardisierten Funktionsbibliothek haben für alle Geräte- und Systemklassen Gültigkeit (s. Kapitel 5.3.3). Die realisierten Strukturierungssichten stellen eine Grundvereinfachung für den Bediener dar. Trotzdem hat dieser die Möglichkeit, durch das flexible Strukturierungskonzept (s. Kapitel 4.3.1) die Erstellung seiner Beschreibungen zusätzlich zu unterstützen. Hierzu kann er weitere Sichtweisen, beispielsweise für Antriebssysteme, nach seinen speziellen Bedürfnissen toolgestützt realisieren (s. Kapitel 6). Bei der standardisierten Klassifikation dieser Arbeit haben bestehende Normen, Richtlinien und Empfehlungen Einfluss genommen.

5.2.2 Fraktale Hilfsfunktionen

An diversen Stellen der modernen Wissenschaft findet der Begriff Fraktal[61] für bestimmte Eigenschaften, Zusammenhänge, Vorgänge und Strukturen Verwendung. WARNECKE versucht durch die „fraktale Fabrik“ die Überlegungen und Erscheinungen in Wissenschaft und Wirtschaft auf einen gemeinsamen Nenner zu bringen [WARNECKE 95]. Das Wort Fraktal ist in der Wissenschaft relativ jung. Es wurde geprägt für die Beschreibung von Strukturen, die mit wenigen, sich wiederholenden Bausteinen zu sehr vielfältigen komplexen, aber aufgabenangepassten Lösungen kommen. Somit sind Rückschlüsse vom einzelnen Teil zum gesamten System und umgekehrt machbar, die bei der Hilfsfunktionalität von Produktionsanlagen und deren unterlagerten Partialsystemen von Nutzen sind.

MANDELBROT[62], der 1975 das Wort Fraktal prägte, stieß bei seinen mathematischen Betrachtungen auf einige Sachverhalte, die nach dem bisherigen wissenschaftlichen Verständnis nicht oder nur schwer zu erklären, anscheinend aber auch zu alltäglich waren, um bislang einmal erwähnt und untersucht zu werden: Ihm fiel auf, dass die Angaben zur Küstenlänge Großbritanniens in verschiedenen Nachschlagewerken stark differierten. Er fand heraus, dass die Abweichungen dadurch zu Stande kamen, dass man den Küstenverlauf in unterschiedlichen Maßstäben vermessen hatte. Je genauer die zu Grunde liegende Karte war, desto länger war die Küste, da mehr Meeresbuchten berücksichtigt werden mussten, die sich zu einer größeren Länge summierten. Mandelbrots Schluss lautete: Wenn man die Karte mit der britischen Küste *unendlich* oft vergrößerte, würde der Küstenverlauf schließlich unendlich lang [MANDELBROT 87].

Wie am Beispiel der Küste zu vermuten ist, setzen sich Fraktale aus disjunkten Teilen derselben Struktur zusammen, wobei jedes Teil dem Ganzen ähnlich ist. In diesem Sinne spricht man von der Selbstähnlichkeit des Fraktals. Jedes Teil

---

[61] Lat. *fractus* = gebrochen

[62] Benoit Mandelbrot, geb. 20.11.1924 in Polen, Mathematiker

des Fraktals hat dann dieselbe Eigenschaft, d. h. die Selbstähnlichkeit von Teilen setzt sich ins Infinitesimale fort.

Man wendet die Theorie der Fraktale aber auch dann an, wenn eine Selbstähnlichkeit nur ungefähr vorhanden ist, d. h. man sie quasi nur durch eine besondere „Brille“ sieht, die die Gesamtsituation des Fraktals „verschwimmen“ lässt:

Betrachtet man beispielsweise die Struktur eines Farnwedels (s. Abbildung 5-9), so erkennt man, dass der gesamte Wedel zu seinen einzelnen Blättern geometrisch ähnlich aufgebaut ist. Diese Ähnlichkeit setzt sich noch einmal fort, wenn man das einzelne Blatt mit den Teilblättern vergleicht; dann aber hört diese Beobachtung auf. Eine Selbstähnlichkeit im Sinne der Mathematik liegt also gar nicht vor, weil die beobachtete Ähnlichkeit nicht bis ins unendlich Kleine fortgesetzt werden kann. Trotzdem lohnt es sich manchmal, Strukturen wie die des Farnwedels als Fraktale zu betrachten oder gar zu erzeugen.

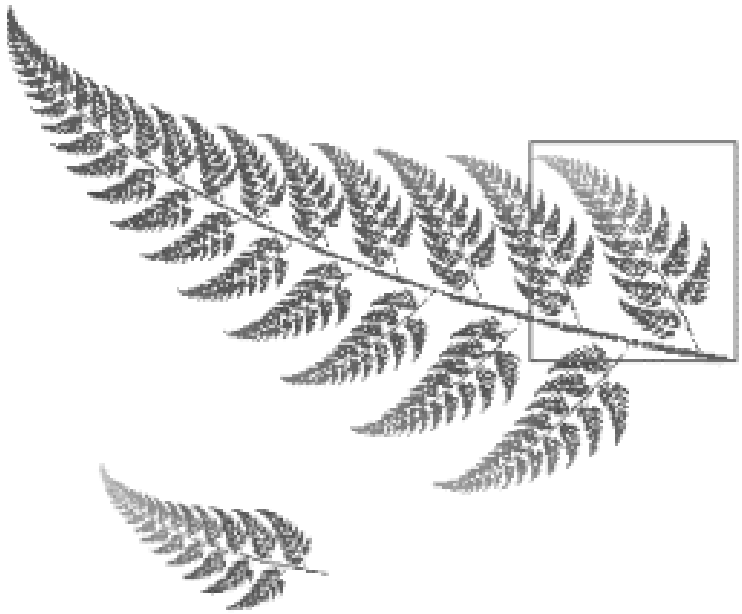


Abbildung 5-9: Selbstähnlichkeit eines Farnblattes

Die überwiegende Zahl der klassischen[63] Fraktale entsteht durch einen iterativen Prozess, in dem bestimmte Elemente (z. B. gewisse Linienstücke oder Flächenteile) zu einem Ausgangsobjekt (= Initiator) hinzugefügt oder entfernt werden. Ein Beispiel ist die nach Helge von Koch benannte Koch-Kurve[64].

---

[63] Die sog. klassischen Fraktale wurden zu Anfang des vorigen Jahrhunderts von bekannten Mathematikern wie Georg Cantor (1872), Guiseppe Peano (1890), David Hilbert (1891), Helge von Koch (1904), Waclaw Sierpinski (1916) oder Gaston Julia (1918) veröffentlicht.

[64] In der Literatur auch als Koch’sche Schneeflocke bezeichnet.

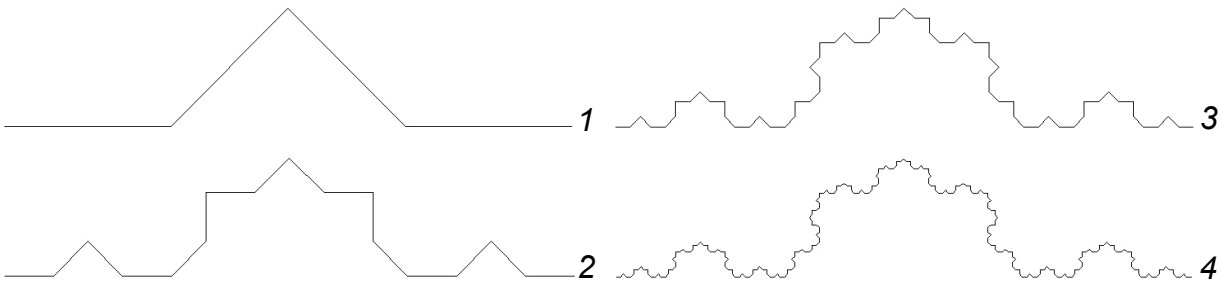


Abbildung 5-10: Bildung des Fraktals Koch-Kurve

Bei Abbildung 5-10 ist der Initiator eine gerade Linie. Diese zerlegt man in drei gleiche Teile. Nun wird das mittlere Drittel der Linie entfernt und durch die Schenken eines gleichseitig rechtwinkligen Dreiecks so ersetzt, dass die Hypotenuse der Länge des entfernten Drittels der Grundlinie entspricht. Im folgenden Schritt wird jede verbleibende gerade Strecke in drei gleiche Teile untergliedert, und die Prozedur wiederholt sich. Es ist ersichtlich, dass mit steigender Iterationszahl die Anzahl der Konstruktionsschritte exponentiell anwächst. Nachdem der Iterationsschritt unendlich oft durchgeführt werden kann, ist die Endfigur selbstähnlich und damit ein Fraktal.

Viele Fraktale besitzen einen gewissen Grad an Selbstähnlichkeit, d. h. sie bestehen aus Teilen, welche dem Ganzen in irgendeiner Weise ähnlich sind. Manchmal kann die Ähnlichkeit schwächer ausfallen als eine strenge geometrische Ähnlichkeit[65]. Sie kann zum Beispiel näherungsweise[66] oder statistisch sein. Fraktale Mengen oder Fraktale können als Objekte interpretiert werden, die bei allen Vergrößerungen (bzw. Verkleinerungen) im Detail in sich selbst übergehen, wenigstens in einem gewissen Größenbereich und für wesentliche Merkmale.

RIEGER geht einen Schritt weiter als WARNECKE und beschäftigt sich über die fraktale Fabrik hinaus mit den fraktalen Strukturen feldbusvernetzter Automatisierungssysteme [RIEGER 95]. Auch wenn die Endlichkeit einer Produktionsanlage nicht die ins Infinitesimale gehende Selbstähnlichkeit der Fraktale zulässt, beschreibt RIEGER das fraktale Automatisierungssystem mit folgenden Eigenschaften:

---

[65] Fraktale mit strikter Selbstähnlichkeit können über Ähnlichkeitsabbildungen mit Kontraktionseigenschaften definiert werden (s. Abbildung 5-10), z. B. Abbildung eines Quadrates auf kleinere Quadrate.

[66] Fraktale mit näherungsweiser (selbst-affiner) Selbstähnlichkeit können über affine Transformationen mit Kontraktionseigenschaften definiert werden, z. B. Abbildung eines Quadrates auf kleinere Rechtecke.

- Jedes Fraktal nimmt Funktionen der Informationsverarbeitung (bestehend aus Nutz- und Koordinationsinformationen) sowie des Bedienens und Beobachtens wahr.
- In Fraktalen werden Informationen verarbeitet und ausgetauscht.
- Jedes Fraktal verfügt durch die Vernetzung aller Fraktale untereinander über ergänzende Informationen kleinerer und größerer Fraktale.
- In Fraktalen werden Informationen verdichtet.

Die kleinste Einheit bilden dabei die prozessnahen Fraktale. Sie enthalten die Aktoren und Sensoren zur Kopplung an den technischen Prozess durch das physikalische Wirkprinzip.

Nachfolgend sind Partialsysteme auf verschiedenen Hierarchieebenen und deren Feldgeräte bezüglich ihrer Hilfsfunktionen zu betrachten.

Definition 5.1

Der fraktale Charakter eines Partialsystems ist gegeben durch die Menge der **fraktalen Hilfsfunktionen,** bezeichnet als **Systemgrundfunktionen**, mit nachfolgenden Eigenschaften:

- Die untereinander ähnlichen Strukturen beinhalten dieselben Hilfsfunktionen zur Bedienung, Projektierung und zum Service.
- Die Menge dieser speziellen Hilfsfunktionen wird als Menge der Systemgrundfunktionen bezeichnet.
- Der fraktale Charakter bezieht sich ausschließlich auf diese Grundfunktionalität des Systems. Diese kann durch beliebige Hilfsfunktionen ergänzt werden.
- Die Abarbeitungslogik bezüglich der ausführenden unterlagerten Funktionen ist für jede fraktale Funktion eindeutig beschrieben.
- Der fraktale Charakter bleibt bei einem Geräteaustausch unverändert.
- Auch die kleinste zu betrachtende Einheit, das intelligente Feldgerät, verfügt über die Systemgrundfunktionen.

Das nachstehende Beispiel soll anschaulich die unmittelbaren Vorteile der fraktalen Hilfsfunktionen verdeutlichen. Es ist eine Anlage zum Lackieren von Autorohkarosserien gegeben. Die vollständige hierarchische Struktur der Lackiererei ist Abbildung 5-11 zu entnehmen. Der Fokus der näheren Betrachtungen liegt auf der Teilanlage zum Lackieren des Grundlackes, speziell auf dem Durchlauftrockner. Dieses Partialsystem ist in Abbildung 5-12 schematisch dargestellt und kann hierarchisch in weitere Partialsysteme gegliedert werden (s. Abbildung 5-11).

Die Annahme, dass die Feldgeräte auf der untersten Hierarchieebene alle spezifizierten Systemgrundfunktionen implementiert haben, dementsprechend konform zum spezifizierten Modell sind, ist bei der vorliegenden Arbeit als obligatorisch zu sehen. Hierauf aufbauend können durch die Fraktalität der Grundfunktionen auf beliebigen Ebenen des hierarchischen Anlagenmodells Bedienobjekte ohne zusätzlichen Implementierungsaufwand generiert werden, die automatisch die Systemgrundfunktionen unterstützen. Die Anlage weist somit auf allen Ebenen des hierarchischen Modells (s. Abbildung 5-11) gleiche Eigenschaften bezüglich der Grundfunktionalität auf, jedes Partialsystem ist im Sinne der Definition zum Gesamtsystem ähnlich.

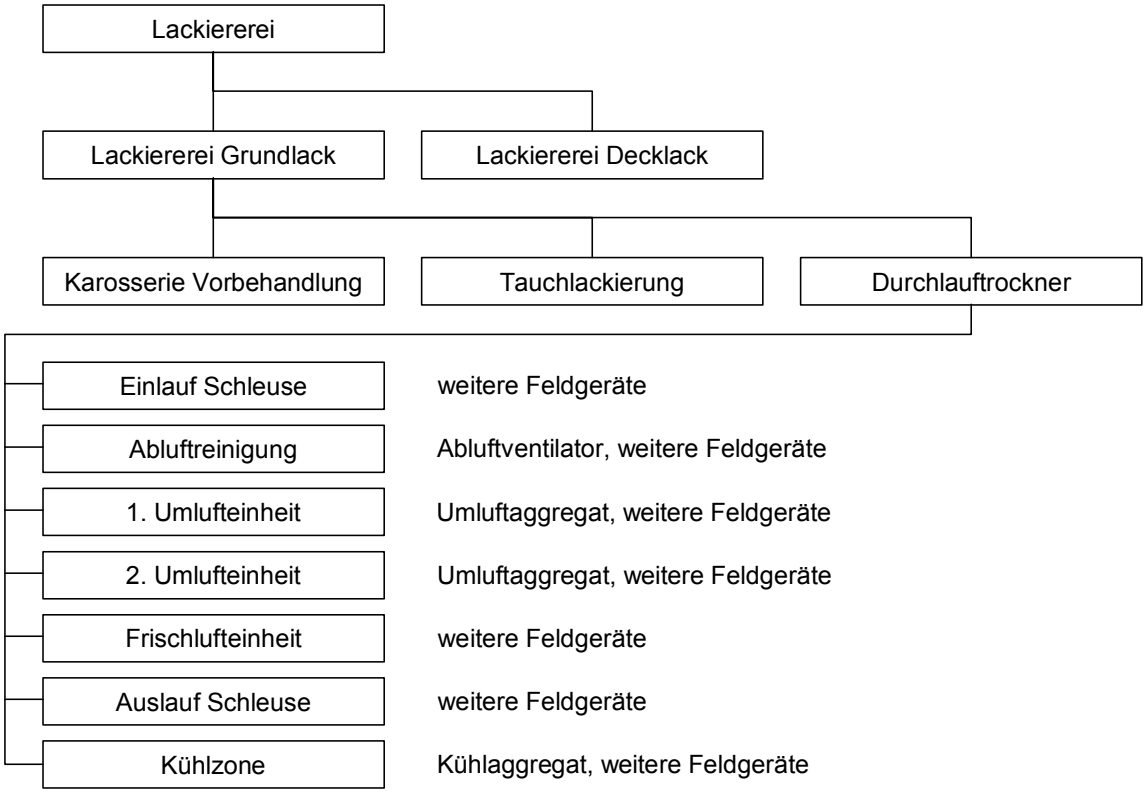


Abbildung 5-11: Hierarchische Struktur der Lackiererei

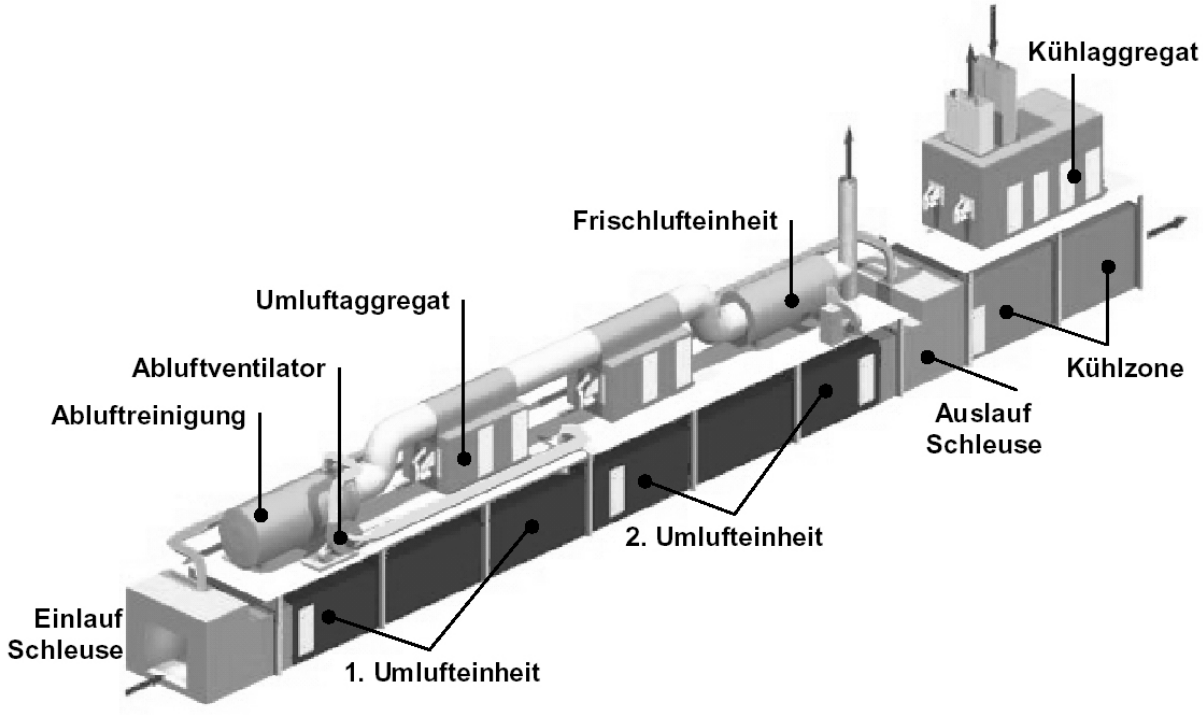


Abbildung 5-12: Schematische Darstellung des Durchlauftrockners [PNO 02C]

Im vorliegenden Fall soll durch einen Mitarbeiter der betrieblichen Instandhaltung geprüft werden, ob eine routinemäßige Wartung wie Kalibrierung eines Sensors etc. an dem Partialsystem Durchlauftrockner in seiner Schicht (entspricht den nächsten 8 Stunden) fällig ist. Hierzu ist die Hilfsfunktion `getMaintenanceData` mit dem entsprechenden Übergabeparameter `Duration = 8` an der Engineering-Station aufzurufen. Der Einstiegspunkt im hierarchischen Anlagenmodell für das Bedienobjekt ist dabei der Durchlauftrockner. Der Funktionsaufruf wird an alle unterlagerten Partialsysteme delegiert[67]. Dieser rekursive Vorgang wird so lange wiederholt, bis auf der Geräteebene die Funktion abgearbeitet ist. Die Diagnosedaten werden über die Partialsysteme zurückgeliefert an das gestartete Bedienobjekt des Durchlauftrockners. Dem Mitarbeiter werden im Ausgabefenster alle Feldgeräte aufgelistet, die einer routinemäßigen Instandhaltung in den folgenden 8 Stunden zu unterziehen sind. Sofern im Bedienobjekt die hierarchische Struktur (Abbildung 5-11) oder die schematische Darstellung (Abbildung 5-12) des Partialsystems visuell enthalten sind, können die Rückgabewerte des Funktionsaufrufes auch dementsprechend umgesetzt werden, sodass diese graphisch der Visualisierung zu entnehmen sind. Somit ist ein schnelles Auffinden der zu wartenden Geräte in der Anlage gewährleistet.

[67] Weiterleitung einer Funktion

Die Menge der Systemgrundfunktionen umfasst hauptsächlich Hilfsfunktionen zur Bedienung und zum Service. Die vollständige Liste aller Systemgrundfunktionen ist der nachfolgenden Tabelle zu entnehmen:

Atomare Hilfsfunktion	***Typ (s. Tab. 5.2)***
backupDeviceData	AND
checkDevice	LIST
getAlarmInfo	LIST
getCountry	LISTALL
getDeviceError	LIST
getDeviceWarning	LIST
getIdentification	LISTALL
getLanguage	LISTALL
getMaintenanceData	LIST
getOperatingTime	LISTALL
gotoFailSafe	AND
login	AND
logout	AND
resetAlarm	AND
resetAllExceptions	AND
resetDevice	AND
resetDeviceError	AND
resetDeviceWarning	AND
resetToDefault	AND
restoreDeviceData	AND
setCountry	AND
setLanguage	AND

Tabelle 5-1: Systemgrundfunktionen

Wie schon erwähnt ist die Grundvoraussetzung, dass diese Funktionen in jedem intelligenten Feldgerät implementiert sind. Die Geräte müssen darüber hinaus aber zusätzliche Hilfsfunktionen enthalten, sodass eine Ausführung der Systemgrundfunktionen überhaupt möglich ist. Um beispielsweise die Systemgrundfunktion `getDeviceError` geräteseitig ausführen zu können ist die zusätzliche Hilfsfunktion `setDeviceErrorParameters` anzuwenden. Hierdurch

ist das Einstellen der Bedingungen für die Auslösung des Fehlers geräteseitig möglich. Erst so kann die Systemgrundfunktion genutzt werden. Eine vollständige Liste aller Gerätegrundfunktionen ist Anhang B zu entnehmen.

Die Art und Weise der Ausführung und Verarbeitung der Übergabe- und Rückgabewerte ist für jede fraktale Hilfsfunktion bekannt und in der Referenzarchitektur spezifiziert. Dadurch kann die Systembeschreibung und darauf aufbauend das Bedienobjekt für die spezielle Gerätekonstellation automatisch generiert werden. Es sind folgende Typen an Systemgrundfunktionen zulässig:

Typ	***Beschreibung***
`AND`	Falls Funktion auf allen Geräten erfolgreich ausgeführt wurde, ist der Rückgabewert „true“ = O.K., ansonsten werden die Gerätenamen, auflösbar durch die Device_IDs, der fehlgeschlagenen Befehlsausführungen aufgelistet.
`LIST`	Auflistung aller Gerätenamen, auflösbar durch die Device_IDs, bei denen der Rückgabewert nicht leer ist.
`LISTALL`	Auflistung aller Gerätenamen, auflösbar durch die Device-IDs, inklusive der Rückgabewerte.

Tabelle 5-2: Unterschiedliche Typen der Systemgrundfunktionen

5.2.3 Polymorphe Hilfsfunktionen

Ein wichtiger Bestandteil des objektorientierten Paradigmas ist der Polymorphismus[68]. Das Konzept des Polymorphismus ist in der Analysephase von untergeordneter Bedeutung und kann erst im Entwurf und in der Implementierung richtig genutzt werden. Der Polymorphismus ermöglicht es, den gleichen Namen für gleichartige Operationen zu verwenden, die auf Objekten verschiedener Klassen auszuführen sind. Der Sender muss nur wissen, dass ein Empfängerobjekt das gewünschte Verhalten besitzt. Er muss nicht wissen, zu welcher Klasse das Objekt gehört. Dieser Mechanismus ermöglicht es, flexible und leicht änderbare Softwaresysteme zu entwickeln [BREUTMANN/BURKHARDT 92].

Bei der Betrachtung der polymorphen Hilfsfunktionen sind zwei verschiedene Sichtweisen möglich. Prinzipiell ist jede Hilfsfunktion auf Geräteebene als polymorph zu sehen, da die Funktionen ausschließlich hinsichtlich ihrer Semantik definiert sind. Die Implementierung in den Feldgeräten erfolgt auf unterschiedlichste Art. Die Komfortfunktionen (s. Kapitel 3.2) sind ebenfalls polymorph, da

[68] Griech.: Vielgestaltigkeit

ein und dieselbe Funktion in verschiedenen Geräten auf Abläufe unterschiedlichster Atomarfunktionen beruhen kann (s. Kapitel 3.2.3).

Die Klassifizierung der Hilfsfunktionen ist an dieser Stelle rein aus der Systemsicht zu sehen. Die fraktalen Funktionen eines Partialsystems benötigen zur Abarbeitung stets semantisch gleiche Funktionen auf den unterlagerten Feldgeräten. Erst hierdurch ist ein Automatismus bei der Erstellung der Systembeschreibung und des Bedienobjektes gegeben. Im Gegensatz hierzu können die verbleibenden Funktionen der standardisierten Bibliothek unterschiedlich realisiert werden.

Definition 5.2

Die Menge der polymorphen Hilfsfunktionen sei definiert über die nachfolgenden Eigenschaften:

Die Ausführung der Hilfsfunktion geschieht durch die Abarbeitung beliebiger Funktionen eines oder auch mehrerer unterlagerter Feldgeräte. Dabei kann es sich sowohl um Hilfsfunktionen als auch um Prozessfunktionen handeln.

Die Wertermittlung erfolgt durch einen frei wählbaren Algorithmus.

Die Erstellung von polymorphen Hilfsfunktionen ist ein Kreativprozess, zu dem ausreichendes Systemwissen notwendig ist. Wie im Beispiel Wärmetauscher (Kapitel 4.1.3) gezeigt wurde, ist die Ausführung einer Hilfsfunktion auf Systemebene sogar durch die Integration von Pozessfunktionen durch den Einsatz der Systemintelligenz möglich. Trotzdem kann der Entwickler bei der Realisierung von polymorphen Hilfsfunktionen, wie beim Konzept eingangs beschrieben, unterstützt werden. So wird beispielsweise durch die Festlegung des auszuführenden Skripttyps, je nachdem ob es sich um die Funktion vom Typ „mapped“, „broadcast“, oder „algorithm“ handelt (s. Kapitel 4.3.3), die passenden Rahmenbedingungen durch das Modell vorgegeben.

Neben der spezifizierten Referenzarchitektur zur einheitlichen Abbildung der Beschreibung eines Partialsystems wird der Entwickler bei der Generierung des Systembedienobjektes (fraktale Hilfsfunktionen) und darüber hinaus bei der Auswahl zusätzlicher Funktionen unterstützt. Hierzu dient die detaillierte Klassifikation, die im nächsten Kapitel beschrieben ist.

5.2.4 Standardstrukturierung

Die entwickelte Ordnungssystematik zur Klassifikation aller Hilfsfunktionen der standardisierten Funktionsbibliothek dient sowohl zur Unterstützung des Geräteherstellers als auch dem Anlagenbauer bei der Erstellung einer Systembeschreibung. Neben diesem primären Ziel der Standardstrukturierung haben Un-

tersuchungen im Rahmen des Verbundprojektes DeKOS gezeigt, dass darüber hinaus ein zusätzlicher Nutzen beim Bediener durch den Einsatz der Ordnungssystematik im Bedienobjekt zu erzielen ist. Als Beispiel sei hier ein Antrieb erwähnt. Dieser kann aufgrund der Anzahl seiner Hilfsfunktionen als komplex bezeichnet werden. Im Gegensatz zu einem Endlagenschalter[69], der mit ca. 4 Hilfsfunktionen beschrieben werden kann, verfügt ein Antrieb im Mittel über 40 Atomarfunktionen und 10 Komfortfunktionen. Das Beispiel Wärmetauscher aus Kapitel 4.1.3 hat gezeigt, dass schon einfache Partialsysteme über viele Hilfsfunktionen verfügen. Diese Menge ist durch den Endanwender zu beherrschen. Durch den Einsatz der Strukturierungssichten auch im Bedienobjekt (s. Kapitel 6) wird eine erhebliche Akzeptanzsteigerung erzielt (s. Kapitel 7.2).

[69] Gehört nicht zur Klasse der intelligenten Feldgeräte.

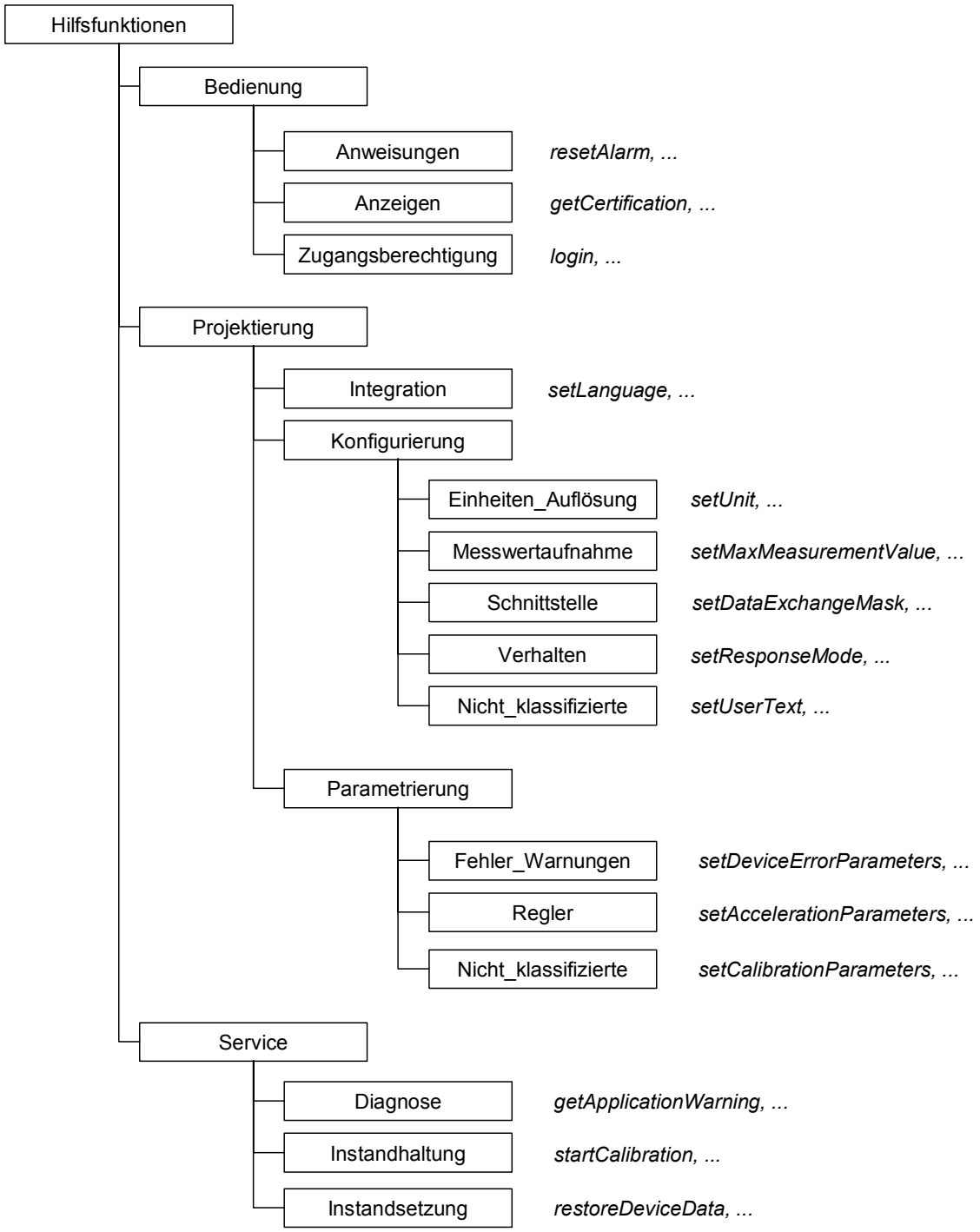


Abbildung 5-13: Standardstrukturierungssicht

Die Klassifikation der Hilfsfunktionen ist Abbildung 5-13 zu entnehmen. Nachdem die Grafik selbsterklärend ist soll nur näher auf die wichtigsten Punkte eingegangen werden. Die Unterklasse Konfigurierung und Parametrierung weist jeweils eine Unterklasse nicht klassifizierter Hilfsfunktionen auf. Dies liegt darin begründet, dass es keinen Sinn macht, für wenige Funktionen eine eigene

Klasse zu definieren. Dies würde den gewünschten Vorteil der Übersichtlichkeit ad absurdum führen. Zu jeder Unterklasse ist ein repräsentierendes Beispiel angegeben. Die vollständige Klassifikation der Funktionsbibliothek ist Anhang C zu entnehmen.

5.3 Formale Beschreibung der Architekturelemente

In den vorhergehenden Kapiteln wurde schon mehrfach belegt, dass XML bis jetzt unangefochten als geeignetes Mittel zur formalen Beschreibung von Informationen angesehen werden kann. Nachdem sich die Sprache in vielen Bereichen, nicht nur in der Automatisierungstechnik, etabliert hat, ist sie zur formalen Beschreibung der Architekturelemente heranzuziehen. Darüber hinaus existieren diverse Softwarekomponenten, wie beispielsweise Standardparser[70], auf deren Basis eine automatische Verarbeitung der Dokumente aufwandsarm realisiert werden kann.

5.3.1 Die deklarative Metasprache XML

In den 70er Jahren wurde von IBM die allererste Markup[71] Sprache namens GML[72] spezifiziert. IBM war damals seiner Zeit weit voraus. Obwohl breite Anwendungen der Spezifikation fehlten, wurde die Sprache in den 80er Jahren weiterentwickelt [LOBIN 00]. Auf Basis dieser Sprache namens SGML[73] wurde 1986 der internationale Standard ISO 8879 verabschiedet [ISO 8879]. SGML gilt als Muttersprache aller Markup Sprachen, so auch von XML. HTML[74] ist eine minimale Definition, die aus dem SGML Standard entstanden ist. Die Befehlswelt von HTML ist dabei sehr bescheiden. Es gibt lediglich eine begrenzte Anzahl an formatierenden Befehlen. Trotz der relativ beschränkten Möglichkeiten hat sich HTML als Seitenbeschreibungssprache im WWW[75] etabliert.

SGML ist eine sehr mächtige Sprache, die wohl gerade wegen ihrer Komplexität nicht den gewünschten Erfolg brachte. XML ist nichts anderes als eine verein-

---

[70] Engl.: Syntaxanalysator, wie beispielsweise der MSXML-Parser von Microsoft. Dieser kostenfrei verfügbare, beziehungsweise im Microsoft Windows Betriebssystem enthaltene Parser kann als DLL (Dynamik Link Library) in das eigene Programm integriert werden und erlaubt mit einfachen Mitteln das Einlesen, Umwandeln, Validieren etc. von XML-Dokumenten.

[71] Engl.: Auszeichnung

[72] Generalized Markup Language

[73] Standardized Generalized Markup Language

[74] Hypertext Markup Language

[75] World Wide Web

fachte Version von SGML. Alle Bestandteile die nicht genutzt worden sind oder besser gesagt aufgrund der Komplexität nicht real nutzbar sind, wurden bei XML weggelassen, ohne dabei die Ausdrucksmöglichkeiten prinzipiell einzuschränken. Eine vom W3C[76] beauftrage Arbeitsgruppe spezifizierte 1998 XML als Untermenge von SGML. Die Spezifikation von XML [W3C 03] ist offen und lizenzkostenfrei verfügbar und bietet inzwischen, mit Hilfe von auf XML basierenden, ergänzenden Spezifikationen, eine umfangreiche Standardfamilie zur Beschreibung, Strukturierung, Transformation und Integration von Daten. Einige dieser Werkzeuge werden in späteren Ausführungen genauer betrachtet.

Obwohl HTML und XML eine gemeinsame Wurzel in der Abstammung haben und sich seitens der Syntax sehr ähnlich sehen, besteht zwischen ihnen ein bedeutender Unterschied. HTML wurde entwickelt, um Daten visuell darzustellen. Dies geschieht durch einen Webbrowser, der das HTML-Dokument interpretiert und die Information entsprechend darstellt. XML dagegen wurde entwickelt um Daten inhaltlich zu beschreiben. Zuallererst sind die wichtigsten Begriffe und Technologien, die in Zusammenhang mit XML Verwendung finden, anhand der Abbildung 5-14 einzuordnen. Es sei darauf hingewiesen, dass die Ausführungen keinen Anspruch auf Vollständigkeit haben, sondern lediglich versuchen, einen Überblick der wichtigsten Fachbegriffe und Standards zu geben, um das Verständnis der formalen Beschreibungen der Architekturelemente in Kapitel 5.3 zu erleichtern. Für detaillierte Informationen sei an dieser Stelle auf die Website des W3C verwiesen [W3C 03].

Ein fundamentaler Vorteil von XML liegt in der rigorosen Trennung von Struktur, Inhalten und der dazugehörigen Formatierung. Diese Trennung vereinfacht das effiziente und flexible Arbeiten mit den Informationen. XML und die begleitenden Standards unterstützen die Trennung, teilweise ist diese sogar notwendig. Im Mittelpunkt stehen:

- Die DTD[77], oder äquivalent hierzu ein XML-Schema.
- Das eigentliche XML-Dokument,
- sowie das XSL[78]-Stylesheet[79].

[76] World Wide Web Consortium
[77] Document Type Definition
[78] Extensible Stylesheet Language
[79] Engl.: Stilvorlage

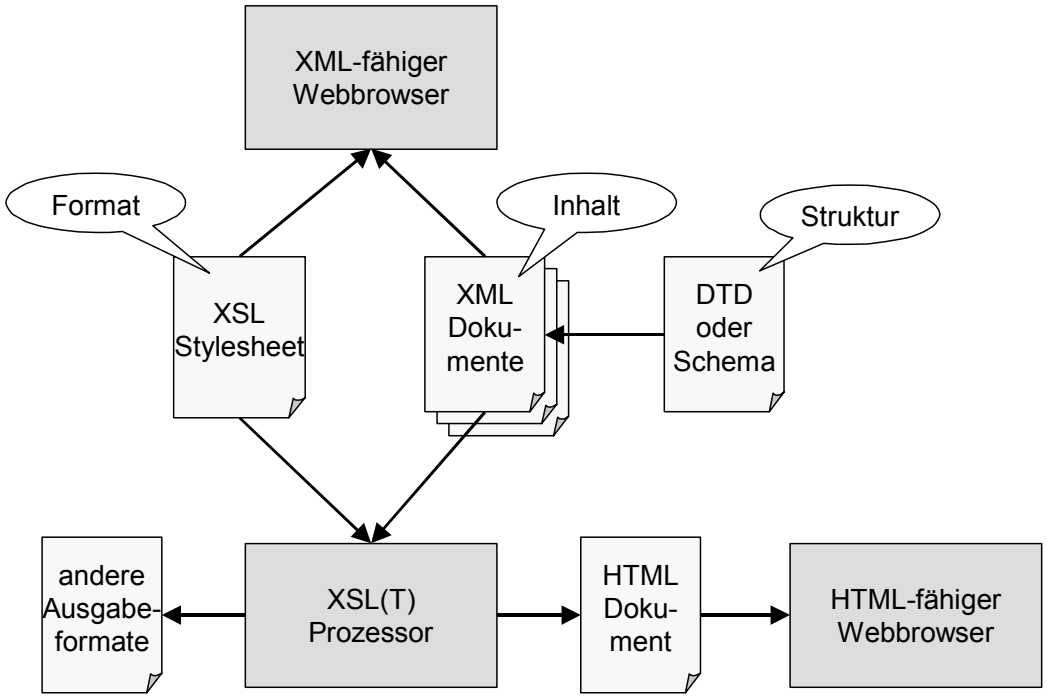


Abbildung 5-14: Schematische Übersicht der XML Familie

Die DTD gibt die Struktur der zu verarbeitenden Informationen vor, eine Erweiterung stellt das Schema dar. Dieses bietet zusätzliche Möglichkeiten auf die an dieser Stelle nicht näher eingegangen wird. Somit kann im vorliegenden Fall eine Referenzarchitektur als Schema definiert werden, anhand dessen eine Validierung bei der Erstellung einer Beschreibung, in Form eines XML-Dokumentes, durch das Softwaretool einfach realisiert werden kann. Die XSL-Stylesheets legen fest, wie die Informationen pro Ausgabeform dargestellt werden sollen. Ein XML-fähiger Browser kann mit Hilfe eines Stylesheets die Informationen eines XML-Dokumentes direkt auf dem Bildschirm darstellen. Als Alternative hierzu kann ein so genannter XSL(T)[80]-Prozessor verwendet werden. Dadurch kann toolgestützt aus den Informationen eines XML-Dokumentes anhand der Anweisungen in dem Stylesheet ein neues Dokument erzeugt werden. Es kann sich beispielsweise um ein HTML-, Word- oder auch pdf-Dokument handeln. Für viele gängige Ausgabeformate sind Softwarekomponenten verfügbar, die ohne zusätzlichen Aufwand im eigenen Softwareprojekt integrierbar sind.

Ein XML-Dokument ist eine Textdatei, die einen *XML-Prolog* sowie mindestens ein *XML-Element* enthält. Der Prolog (s. Abbildung 5-15) besteht dabei aus der Versionsangabe und dem Attribut ENCODING.

[80] XSL-Transformation

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Abbildung 5-15: Prolog eines XML-Dokuments

Es gibt den Zeichensatz an, mit welchem das Dokument behandelt werden soll. Als Attributswerte sind zu wählen:

Encoding:	***Erklärung:***
UTF-8	Internationaler Zeichensatz mit 8 Bit Breite
UTF-16	Internationaler Zeichensatz mit 16 Bit Breite
ISO-8859-1	ISO-Zeichensatz für westeuropäische Sprachen

Tabelle 5-3: Mögliche Zeichensätze eines XML-Dokuments

Optional kann eine *Dokumententypdeklaration* oder ein Verweis auf eine DTD folgen, die Erläuterungen hierzu kommen später. Das Beispiel beinhaltet, dass es sich um ein XML-Dokument der Version 1.0 handelt, das mit dem Zeichensatz gemäß ISO 8859-1 kodiert ist. Die aktuelle Version 1.1 liegt seit kurzem vor [W3C 03]. Die Spezifikation der vorliegenden Arbeit erfolgte auf dem Stand der Version 1.0 Second Edition.

Wie schon erwähnt, beinhaltet jedes XML-Dokument mindestens ein *XML-Element*. Die Grundregel zur Bildung eines Elements besteht in der Anordnung von *Content* zwischen einem *Start-Tag* und einem *End-Tag*, bei dem es sich um reinen Text handelt. Die *Element-Names* sind frei wählbar und erscheinen sowohl im Start-Tag als auch im End-Tag nach folgendem Muster:

```
<Contact>Personenliste</Contact>
```

Abbildung 5-16: XML-Element mit Textinhalt

Abbildung 5-16 zeigt ein frei definiertes Element namens `Contact`, dessen Inhalt der Text `Personenliste` ist. Bei der Verschachtelung von Elementen ist darauf zu achten, dass der End-Tag des jeweils inneren Elements vor dem End-Tag des umschließenden Elements erscheint. Ein Beispiel hierzu ist Abbildung 5-17 zu entnehmen.

```
<Contact>
   <Name>Heiko Meyer</Name>
   <Mail>meyer@itm.tum.de</Mail>
</Contact>
```

Abbildung 5-17: Verschachtelung von XML-Elementen

Zusätzlich zum Content können Elemente *Attribute* besitzen, die im Start-Tag des Elements angegeben sind. Der Attributwert ist dabei in Anführungszeichen zu setzten (siehe hierzu auch Abbildung 5-15).

```
<Contact>
   <Name>Heiko Meyer</Name>
   <Phone type="office">+498928916446</Phone>
   <Phone type="private"></Phone>
   <Mail>meyer@itm.tum.de</Mail>
</Contact>
```

Abbildung 5-18: XML-Element mit Attribut

Attribute werden zur näheren Spezifikation eines Elements eingesetzt. Die Namen der Attribute sind wie die der Elemente frei wählbar. Das Beispiel aus Abbildung 5-18 bringt zum Ausdruck, dass durch das Attribut type das Element Phone genauer unterschieden werden kann. Es sind auch *Leerelemente*, wie bei type="private" zulässig.

Das *Wurzelement* unterscheidet sich von allen anderen Elementen des Dokuments dahingehend, dass es an den Prolog folgen muss, nur dort verwendet werden darf und alle anderen Elemente und Inhalte umschließt.

Optional kann ein XML-Dokument im Prolog eine zuvor erwähnte Dokumententypdeklaration aufweisen, die eine DTD beinhaltet oder referenziert. Innerhalb dieser ist es unter anderem möglich, die im Dokument erlaubten Element- und Attributsnamen festzulegen, sowie die Verschachtelungsstruktur der Elemente vorzugeben. Werden diese Regeln durch die Elemente im Hauptteil des Dokuments verletzt, spricht man von einem *non-valid* Dokument.

Die DTD ermöglicht so die Vorgabe von Namensräumen und Strukturen. In der Regel werden die DTD nicht im Dokument selber sondern in einer externen Datei abgelegt, damit mehrere XML-Dokumente auf dieselbe DTD verweisen können. Wie zuvor beschrieben, kann hierdurch ein Parser beliebige XML-Dokumente bezüglich der Elemente, Attribute und Strukturen gegenüber einer DTD validieren.

Ein XML-Schema hat aus logischer Sicht sehr viel mit einer DTD gemein. Beide dienen zur Beschreibung der Struktur eines XML-Dokuments sowie der Formulierung von Wertebereichseinschränkungen von Elementinhalten und

Attributen. Darüber hinaus ermöglicht ein Schema die Zuweisung einfacher Datentypen, wie z. B. integer, die Erzeugung abgeleiteter Datentypen wie z. B. integer im Bereich von 1 bis 10 oder auch der Definition komplexer Datentypen. Auch ermöglicht ein Schema die genaue Beschränkung der Instanzen eines Elements. Die umfangreichen Möglichkeiten bei der Verwendung eines Schemas sind der *XML-Schema Definition Language* zu entnehmen [W3C 03].

5.3.2 XML-Schema StrucRefArchitecture

Die nachfolgenden Kapitel gehen auf die formale Beschreibung der zuvor spezifizierten Modelle ein, zuvor soll allerdings ein kurzer Überblick der Beziehungen zwischen den Partialmodellen gegeben werden (s. Abbildung 5-19). Das in Kapitel 4.3.1 spezifizierte Klassenmodell zur Darstellung der Ordnungssystematik der Hilfsfunktionen wird in Form des XML-Schemas *StrucRefArchitecture* abgebildet. Auf Basis dessen können die in Kapitel 5.2 definierten Strukturierungen dargestellt werden. Eine Erweiterung der Sichten durch den Benutzer ist ohne größeren Aufwand toolgestützt möglich (s. Kapitel 6). Hierzu ist eine Validierung des XML-Dokumentes durch das Schema gegeben.

Die Referenzarchitektur zur Beschreibung der Hilfsfunktionen eines Partialsystems ist in Kapitel 4.3.3 spezifiziert worden. Dieses Klassenmodell wird in Form des XML-Schemas *SysRefArchitecture* abgebildet. Es dient zur Instanzierung der Partialsystembeschreibungen. Dabei geben die Informationen der Strukturierungssichten lediglich eine geeignete Darstellung bei der Auswahl der Hilfsfunktionen wieder. Im abgeleiteten XML-Dokument *DDD.xml* ist die Ordnungssystematik nicht enthalten.

Auf Basis der vollständigen Partialsystembeschreibung kann, wie in Kapitel 6 noch gezeigt wird, das Bedienproxy des Partialsystems generiert werden. Zusammen mit der Benutzerschnittstelle bilden diese das Bedienobjekt, eingebettet in einer Rahmenapplikation. Das GUI wird beim Start dynamisch aus der geparsten Information der Systembeschreibung erzeugt. Zusätzlich wird auch die Strukturierung der Funktionen im Benutzerdialog abgebildet. Somit müssen dem GUI beim Start die XML-Dokumente zur Verfügung stehen. Die unmittelbaren Vorteile einer dynamischen Erzeugung im Gegensatz zu einer statischen Implementierung werden in Kapitel 6 näher erläutert.

Im Folgenden werden die Grundkonzepte des XML-Schemas zur formalen Notation des Strukturierungskonzeptes vorgestellt. Die vollständige Beschreibung ist Anhang D zu entnehmen und ist im Gegensatz zur DTD wie alle Schemas in XML notiert. Anhand von Auszügen des Dokumentes ist die Umsetzung des zuvor spezifizierten Klassenmodells nachvollziehbar.

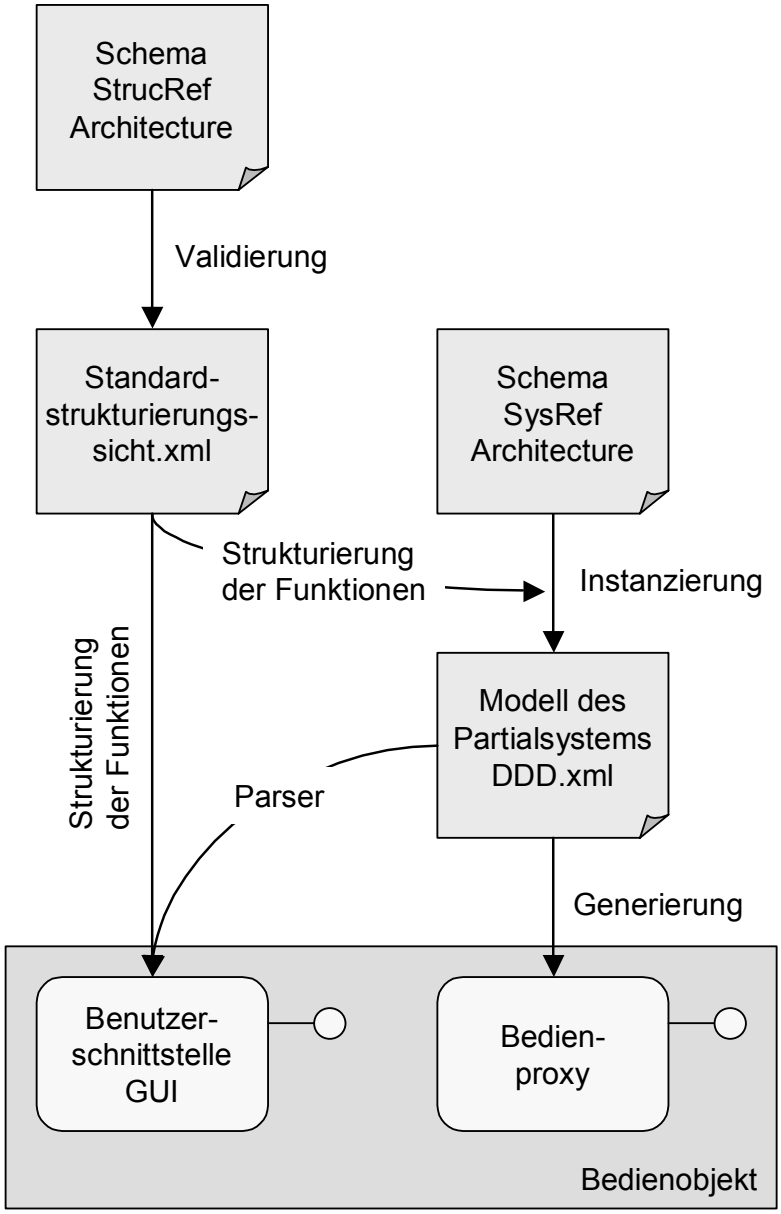


Abbildung 5-19: Beziehungen zwischen den Partialmodellen

Das Vokabular für die Erstellung eines XML-Schemas ist durch den Namensraum *http://www.w3.org/2001/XMLSchema.xsd* festgelegt. Über das Präfix *xsd* ist dieser zu referenzieren. Das Wurzelelement ist durch den Namen *schema* zu kennzeichnen, sodass sich nachfolgendes oberstes Element ergibt:

```
<xsd:schema targetNamespace=http://www.itm.tum.de/2001/adml
            xmlns=http://www.itm.tum.de/2001/adml
            xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

Abbildung 5-20: Definition des Wurzelements der StrucRefArchitecture

Zu unterscheiden sind die Elemente ohne Präfix, die unter *http://www.itm.tum.de/2001/adml*[81] definiert sind und die Elemente mit Präfix xsd, die über die XML-Spezifikation referenziert werden. Die identifizierten Klassen aus Kapitel 4.3.1 werden nachfolgend in Form von Typdefinitionen abgebildet.

```
<xsd:element name="STRUCTURES">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="FILEINFO" type="FileInfoType"
                   minOccurs="0" maxOccurs="1"/>
      <xsd:element name="VIEW" type="ViewType"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="NAME" type="xsd:string"
                   use="required"/>
  </xsd:complexType>
</xsd:element>
```

Abbildung 5-21: Definition der Klasse Structures

Jedes XML-Dokument das konform zum Schema StrucRefArchitecture erstellt wird, enthält diverse Strukturierungssichten. Dementsprechend wird das Hauptelement des Schemas repräsentiert durch die Abbildung der Klasse `STRUCTURES`. Die Definition beginnt mit dem `Start-Tag xsd:complexeType`, das besagt, dass es sich hierbei um einen komplexen Datentyp handelt, der Unterelemente sowie Attribute enthält. Die nachfolgende Definition wird gefasst von dem `Tag xsd:sequence`. Hierdurch ist festgelegt, dass die Abbildung der Unterelemente in einer XML-Datei nach der Reihenfolge ihrer Definition im korrespondierenden Schema zu erfolgen hat. Das Unterelement `FILEINFO` ist vom Typ `FileInfoType`, der die Klasse `FileInfo` explizit definiert. Durch die Werte der Attribute `minOccurs` und `maxOccurs` ist festgelegt, dass dieses Element optional innerhalb einer Instanz von `STRUCTURES` erscheinen kann. Gleiches gilt für die Definition des Elements `VIEW` mit dem wesentlichen Unterschied, dass die Anzahl der enthaltenen Sichten beliebig sein kann.

Abschließend wird das Attribut `NAME` des zu definierenden Elements festgelegt. Es wird der Standarddatentyp `string` aus der XML-Schema Spezifikation gewählt. Durch `use="required"` ist beschrieben, dass es sich um ein zwingend erforderliches und nicht optionales Attribut handelt.

Folgende weitere Klassen des Modells der StrucRefArchitecture werden aufgrund von Vererbung, Mehrfachverwendung etc. als explizite Typen definiert:

[81] Der gesamte Namensraum wird in der ADML zusammengefasst und steht für Automation Device Markup Language.

`FileInfo`, `Structure`, `Description` und `FunctionGroup`. Zu den detaillierten Ausführungen sei an dieser Stelle auf den Anhang verwiesen.

Konform zum beschriebenen Klassenmodell der StrucRefArchitecture in einem XML-Schema ist eine Standardstrukturierungssicht als XML-Dokument erstellt worden. Diese bildet alle definierten Sichten und Funktionsklassen aus Kapitel 5.2 ab. Das vollständige XML-Dokument ist Anhang E zu entnehmen.

5.3.3 XML-Schema SysRefArchitecture

Wie in Kapitel 4.3.3 eingehend erläutert, basiert das entworfene Klassenmodell SysRefArchitecture als Referenzarchitektur für die Systembeschreibungen, teilweise auf der vorhandenen DRA. Dementsprechend können auch Klassen des vorhandenen XML-Schemas der DRA wieder verwendet werden. Die Erweiterung erfolgt über die Klasse F2Function. Dabei ist das XML-Schema derart gestaltet, dass die zusätzlichen Klassen zur Systembeschreibung nicht notwendigerweise in den Instanzen vorhanden sein müssen, sondern optional hinzugefügt werden können. So sind auch die Gerätebeschreibungen basierend auf der DRA mit der SysRefArchitecture validierbar.

Nachfolgend werden die Grundkonzepte zur formalen Notation der Referenzarchitektur zur Systembeschreibung vorgestellt. Anhand von Auszügen des Dokumentes ist die Umsetzung des in Kapitel 4.3.3 spezifizierten Klassenmodells nachvollziehbar. Dabei wird nur auf die wesentlichen Klassen bezüglich der Systembeschreibung eingegangen. Die vollständige Beschreibung ist Anhang F zu entnehmen.

```
<xsd:complexType name="f2functionType">
  <xsd:complexContent>
    <xsd:extension base="controlfunctionType">
      <xsd:sequence>
        <xsd:element name="SCRIPT" type="scriptType"
                     minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="BASICFUNCTION">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="AND"/>
            <xsd:enumeration value="LIST"/>
            <xsd:enumeration value="LISTALL"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="DEKOSNR" type="xsd:ID"
                     use="required"/>
      <xsd:attribute name="NAME" type="f2functionnameType"
                     use="required"/>
      <xsd:attribute name="FUNCTYPE">
```

```
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="GET"/>
            <xsd:enumeration value="SET"/>
            <xsd:enumeration value="COMMAND"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Abbildung 5-22: Definition der Klasse F2Function

Das Wurzelelement gestaltet sich äquivalent zu dem der StrucRefArchitecture und ist nicht näher zu erläutern. Neben den Interfaces, den Komfortfunktionen etc. enthält jedes System Atomarfunktionen. Wie eingangs erwähnt bilden diese die Schnittstelle zu den unterlagerten Geräten.

Die Klassendefinition (s. Abbildung 5-22) beginnt mit dem Text `Start-Tag xsd:complexType`, der besagt, dass es sich hierbei um einen komplexen Datentypen handelt. Dieser beinhaltet sowohl Elemente als auch Attribute. Das Element `xsd:extension` als Unterelement der Typdefinition bringt zum Ausdruck, dass die Klasse `F2Function` als Erweiterung der Basisklasse `ControlFunction` repräsentiert durch die Typdefinition `controlfunctionType` zu verstehen ist. Im Beispiel wird die Basistypdefinition um das Element `SCRIPT` und die vier Attribute `BASICFUNCTION`, `DEKOSNR`, `NAME` und `FUNCTYPE` erweitert. Dabei werden die beiden Attribute `BASICFUNCTION` und `FUNCTYPE` durch eine anonyme Typdefinition spezifiziert, was durch das Fehlen eines Typnamens zum Ausdruck kommt. Die anonymen Typen werden durch Aufzählungen von vordefinierten Werten des eingebauten Typs `xsd:string` definiert. Das Element `xsd:restriction` spezifiziert den Ausgangsdatentyp, die Elemente `xsd:enumeration` beinhalten die Aufzählungswerte.

```
<xsd:complexType name="scriptType">
  <xsd:sequence>
    <xsd:element name="STEP" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:attribute name="COMMAND" type="xsd:string"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="NAME" type="xsd:string"/>
  <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
  <xsd:attribute name="TYPE">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
```

```
        <xsd:enumeration value="MAPPED"/>
        <xsd:enumeration value="BROADCAST"/>
        <xsd:enumeration value="ALGORITHM"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
```

Abbildung 5-23: Definition der Klasse Script

Die Klasse `Script` vom Typ `complexType` beinhaltet das Elemente `STEP`, sowie die Attribute `NAME`, `DESCRIPTION` und `TYPE` (s. Abbildung 5-23). Bei dem Element `STEP` handelt es sich um die Codezeilen des Skripts vom Typ `xsd:string`. Der Skripttyp wird über das Attribut `TYPE` dargestellt. Auch hierbei handelt es sich um eine anonyme Typdefinition, definiert durch eine Aufzählung vordefinierter Werte vom eingebauten Typ `xsd:string`.

```
<xsd:complexType name="subdeviceType">
  <xsd:attribute name="NAME" type="xsd:string"
                 use="required"/>
  <xsd:attribute name="GUID" type="xsd:string"/>
  <xsd:attribute name="DEVICE ID" type="xsd:ID"
                 use="required"/>
</xsd:complexType>
```

Abbildung 5-24: Definition der Klasse SubDevice

Die Klasse `SubDevice` beinhaltet die unterlagerten Feldgeräte eines Partialsystems (s. Abbildung 5-24). Dabei wird das Attribut `NAME` aus der Gerätebeschreibung übernommen. Nachdem ein Gerätetyp in einer Anlage auch mehrfach enthalten sein kann, ist zusätzlich zur eindeutigen Identifikation im Partialsystem das Attribut `DEVICE_ID` vergeben. Neben den zwingend erforderlichen Attributen kann optional für die Implementierung das Attribut `GUID` zur Identifikation des realen Geräteproxys in die Systembeschreibung aufgenommen werden.

```
<xsd:complexType name="vardeclarationType">
  <xsd:attribute name="VAR" type="xsd:string"/>
  <xsd:attribute name="DEVICE ID" type="xsd:ID"/>
  <xsd:attribute name="DEKOSNR" type="xsd:ID"/>
  <xsd:attribute name="DIRECTION">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="IN"/>
        <xsd:enumeration value="OUT"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
```

Abbildung 5-25: Definition der Klasse VarDeclaration

Zum leichteren Umgang mit den Funktionswerten im Programmskript werden in diesem interne Variablen verwendet, die über die Klasse `VarDeclaration` festzulegen sind (s. Abbildung 5-25). Dabei ist der Name über das Attribut `VAR` vom Typ `xsd:string` bestimmt. Der Verweis auf das unterlagerte Feldgerät erfolgt über die interne `DEVICE_ID`. Die aufzurufende Funktion ist über die eindeutige `DEKOSNR` vom Typ `xsd:ID` referenziert. Zusätzlich wird durch das anonyme Attribut `DIRECTION` deklariert, ob es sich um einen Ein- oder Ausgabewert der Funktion handelt.

```
<xsd:simpleType name="f2functionnameType">
  <xsd:restriction base="xsd:NMTOKEN">
    <xsd:enumeration value="browseApplication"/>
    <xsd:enumeration value="backupDeviceData"/>
    <xsd:enumeration value="restoreDeviceData"/>
    …
  </xsd:restriction>
</xsd:simpleType>
```

Abbildung 5-26: Ausschnitt aus der Klasse F2FunctionName

Durch die Analyse verschiedener Partialsysteme aus Anlagen der Verfahrens- und Fertigungstechnik wurde der Namensraum der Spezifikation hinsichtlich der Vollständigkeit zur Beschreibung der Hilfsfunktionalität überprüft. Dabei wurden neue Atomarfunktionen wie `setEventReaction`, `backupDeviceData` etc. definiert und in die Referenzarchitektur aufgenommen (s. Abbildung 5-26). Die Klasse ist dabei relativ einfach aufgebaut. Sie besteht ausschließlich

aus der Typedefinition, einer Aufzählung von Werten des Typs `xsd:NMTOKEN`, des so genannten Name-Token[82].

```
<xsd:element name="DeKOSDEVICE">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="FILEINFO" type="fileinfoType"/>
      <xsd:element name="SUBDEVICE" type="subdeviceType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="VARDECLARATION"
                   type="vardeclarationType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="PRODFUNCTION" type="prodfunctionType"
                   maxOccurs="unbounded"/>
      <xsd:element name="F2FUNCTION" type="f2functionType"
maxOccurs="165"/>
      …
      </xsd:sequence>
      <xsd:attributeGroup
ref="deviceidentificationAttributes"/>
    </xsd:complexType>
  </xsd:element>
```

Abbildung 5-27: Definition der Klasse DeKOSDevice

Nachdem alle Klassen und Attribute des Modellierungskonzeptes aus Kapitel 4 durch Typdefinitionen abgebildet sind, ist es erforderlich ein Wurzelelement zu definieren, das ein konkretes Partialsystemmodell umfasst. Das im vorliegenden Fall spezifizierte Element trägt in Anlehnung an die DRA den Namen `DeKOS-Device`, obwohl es sich auch um die Beschreibung eines Partialsystems handeln kann. Innerhalb des Wurzelelementes werden die Modellelemente mit den entsprechenden Multiplizitäten deklariert. Direkt unterhalb des Wurzelelementes sind alle Funktionen und Interfaces, die ein System besitzt, zu erstellen.

Zur Darstellung der vollständigen Referenzarchitektur sei an dieser Stelle auf den Anhang F verwiesen. Konform zum formal beschriebenen Klassenmodell der SysRefArchitecture in Form des an dieser Stelle beschriebenen XML-Schemas wurde ein XML-Dokument erstellt. Dieses kann als Beschreibung eines Metasystems bezeichnet werden, das über alle spezifizierten Interfaces, Atomarfunktionen, etc. verfügt. Dabei werden auch die Grundfunktionen als solche gekennzeichnet und über das Attribut `BASICFUNCTION` die Art der booleschen Verknüpfung für jede dieser atomaren Grundfunktionen festgelegt. Auf einen Abdruck des XML-Dokumentes im Anhang wird aufgrund der Länge ver-

[82] Im Gegensatz zu xsd:string handelt es sich hierbei um eine Kombination von Buchstaben, Ziffern sowie verschiedenen Interpunktionszeichen, allerdings ohne Sonder- und Leerzeichen.

zichtet. Die Eigenschaften der Funktionen können aber Tabelle 5-2 entnommen werden.

5.4 Zusammenfassung

Zu Beginn des Kapitels 5 wurde die Ordnungssystematik zur Klassifikation der standardisierten Hilfsfunktionen festgelegt. Hierzu wurden in einem ersten Schritt bestehende Ansätze analysiert. Nach der Einführung eines geeigneten Klassifikationsverfahrens folgte die Beschreibung der definierten Funktionsklassen und deren Eigenschaften. Die Einführung verschiedener *Strukturierungssichten* (s. Kapitel 5.2.1) auf ein und dieselbe Menge an Funktionen erwies sich als geeignet, um die Anforderungen bezüglich des Engineerings aus Kapitel 2 zu erfüllen.

Um das Ziel der automatischen Generierung eines Bedienobjektes auf Partialsystemebene zu erreichen, wurden u. a. fraktale Hilfsfunktionen definiert. Diese als Systemgrundfunktionen benannten Atomarfunktionen beinhaltet jedes Partialsystem, vorausgesetzt, die unterlagerten Feldgeräte haben diese Hilfsfunktionen implementiert. Die Abarbeitungslogik ist dabei für jede einzelne Funktion festgelegt.

Der zweite Teil des Kapitels beschäftige sich mit der formalen Beschreibung der in Kapitel 4 spezifizierten Klassenmodelle. Die Metasprache XML stellte sich als geeignet heraus um die formale Notation durchzuführen. Zum besseren Verständnis für den Leser wurde eine kurze Einführung in die deklarative Metasprache XML gegeben. Die spezifizierten Klassenmodelle wurden durch XML-Schemata abgebildet. Die Beziehungen der Partialmodelle und Beschreibungen wurden bezüglich der Validierung und Instanzierung grafisch festgelegt.

Die hierdurch vorhandene formale Spezifikation der Partialmodelle StrucRefArchitecture und SysRefArchitecture ermöglicht die modellbasierte Generierung einheitlicher Bedienobjekte für beliebige intelligente mechatronische Systeme sowohl aus dem Bereich der Verfahrens- als auch der Fertigungstechnik. Dabei findet der gesamte Engineeringprozess, angefangen bei der Erstellung der passenden Gerätebeschreibungen bis hin zur Abbildung der funktionalen Zusammenhänge im mathematischen Sinne, durch die Modellierung Unterstützung.

Kapitel 6 zeigt im Rahmen eines durchgängigen Anwendungsbeispiels die konkrete Ausführung, um das Verfahren zur modellbasierten Generierung zu validieren. Dabei wird nicht nur auf die Erstellung der Geräte- und Systembeschreibungen eingegangen, sondern auch die prototypische Unterstützung durch Softwaretools erläutert und die Softwarearchitektur der verwendeten Proxy-Technologie näher betrachtet.

6 Prototypische Anwendung des Modellierungskonzeptes

Das Kapitel der prototypischen Anwendung gestaltet sich dreiteilig. Der Lösungsweg der vorliegenden Arbeit beinhaltet sowohl den gesamten Engineering-Prozess als auch die geeignete Modellbildung zur Realisierung eines Bedienobjektes für Partialsysteme. Zuerst wird auf die Unterstützung durch das Softwaretool zur Erstellung DeKOS-konformer Beschreibungen eingegangen. Nach einer Einführung in die Grundlagen zur Software-seitigen Umsetzung der Bedienobjekte gibt der dritte Teil des Kapitels ein durchgängiges Beispiel aus der Verfahrenstechnik wieder.

6.1 Tool zur Erstellung DeKOS-konformer Beschreibungen

6.1.1 Assistentengestützte Generierung einer DDD

Im Rahmen des Verbundprojektes DeKOS ist am itm ein prototypisches Softwaretool in Java zur Erstellung von DeKOS-konformen Gerätebeschreibungen entstanden. Dieses Tool wurde bezüglich eines verbesserten Engineerings weiterentwickelt und durch Features zur Erstellung von Partialsystembeschreibungen ergänzt. Bei dem Redesign ist die Wahl der Programmiersprache zugunsten von Visual Basic V6.0 (VB) getroffen worden. Die Realisierung komplexer graphischer Benutzeroberflächen beispielsweise mit Drag&Drop-Techniken ist hierdurch mit überschaubarem Aufwand gegeben.

Das Tool bietet auf Basis der spezifizierten Modelle folgende Eigenschaften:

- Erstellung von Beschreibungen für beliebige Feldgeräte.
- Erstellung von Gerätebeschreibungen mit entsprechenden Grundfunktionen zur Nutzung der fraktalen Hilfsfunktionen auf Partialsystemebene.
- Erstellung von Beschreibungen für Partialsysteme.
- Erstellung der Abläufe für Komfortfunktionen (s. Kapitel 3.2.2).
- Generierung des Funktionsgerüstes für eine VB-Implementierung (Proxy auf Basis der FDT-Technologie).
- Erstellung zusätzlicher Strukturierungssichten (s. Kapitel 6.1.2).

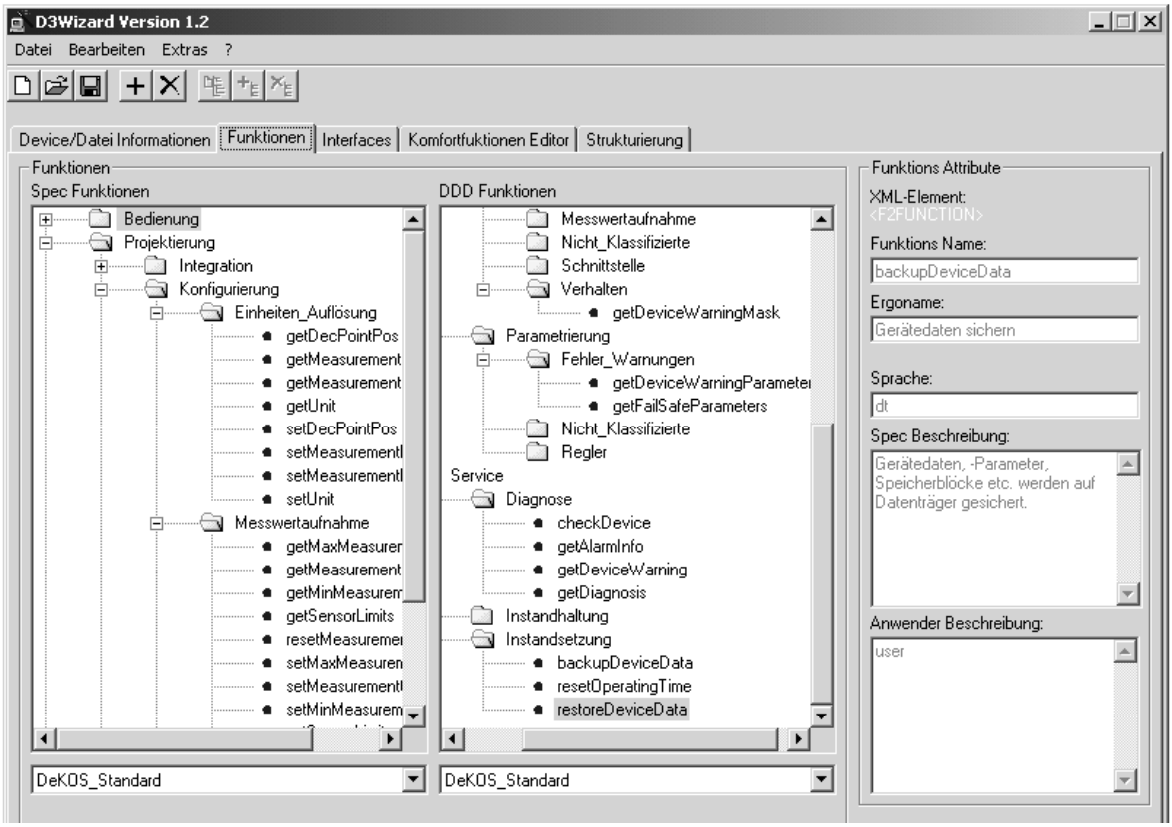


Abbildung 6-1: Benutzeroberfläche des D3Wizards

Abbildung 6-1 zeigt die Benutzeroberfläche des Tools D3Wizard. Mittels einfacher Aktionen, gemäß dem Windows Look&Feel, ist ohne Programmierkenntnisse die Erstellung der Beschreibungen möglich. Über die verschiedenen „Karteikarten“ sind die einzelnen Schritte einfach ausführbar. Es sind zunächst die Geräte-/Systemdateiinformationen zu bestimmen. Bei der Auswahl der benötigten standardisierten Hilfsfunktionen der Bibliothek kann der Benutzer jederzeit über die Combobox zwischen den angebotenen Strukturierungssichten wechseln (s. Abbildung 6-2). Die Funktionen werden aus der Funktionsbibliothek (linker Fensterbereich) in die zu erstellende Beschreibung gezogen (rechter Fensterbereich).

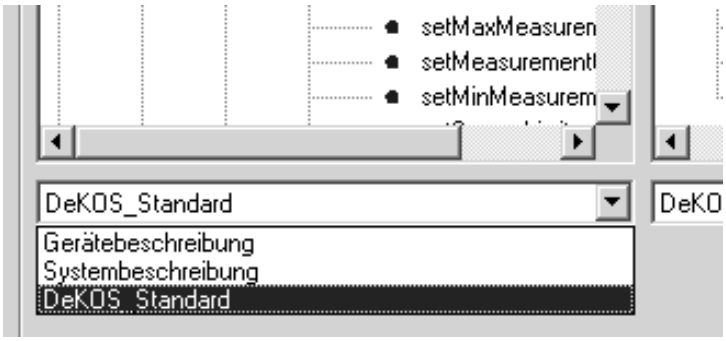


Abbildung 6-2: Auswahl der Strukturierungssicht über die Combobox

Die Auswahl der benötigten Interfaces zur Abbildung der Benutzerrollen und das Editieren der Komfortfunktionen ist auf weiteren Karteikarten enthalten. Unterstützt wird zudem die Erstellung herstellerspezifischer Prozessfunktionen. Letztendlich exportiert das Tool eine fertige Beschreibung (DDD) als XML-Dokument, konform zur SysRefArchitecture, validierbar über das spezifizierte XML-Schema. Aus dieser DDD kann das Funktionsgerüst, eine html- oder Word-Datei zur Dokumentationserstellung durch Standardverfahren erstellt werden.

6.1.2 Erstellung zusätzlicher Strukturierungssichten

Auf die Erstellung zusätzlicher Strukturierungssichten soll gesondert eingegangen werden, da dies den gesamten Engineering-Prozess und die Benutzung des fertig gestellten Bedienobjektes für den Endanwender wesentlich beeinflusst. Der Bedarf nach Strukturierung der Funktionsbibliothek wurde in Kapitel 4.2.3 belegt. Hierzu wurden bei der formalen Beschreibung in Kaptiel 5.2 drei Strukturierungssichten erarbeitet. Untersuchungen im Rahmen des Verbundprojektes DeKOS haben ergeben, dass es für den involvierten Personenkreis durchaus sinnvoll und hilfreich sein kann, nach den eigenen Vorstellungen die Hilfsfunktionen zusätzlich zu strukturieren. Dies betrifft sowohl die Erstellung von Beschreibungen als auch die angebotene Funktionalität im lauffähigen Bedienobjekt. Auf Basis der spezifizierten StrucRefArchitecture in Form eines XML-Schemas ist es durch den D3Wizard möglich, eigene Klassifikationen der Funktionsbibliothek als Strukturierungssichten zu realisieren. Diese kann dann sowohl im D3Wizard als auch im GUI des Bedienobjektes Verwendung finden.

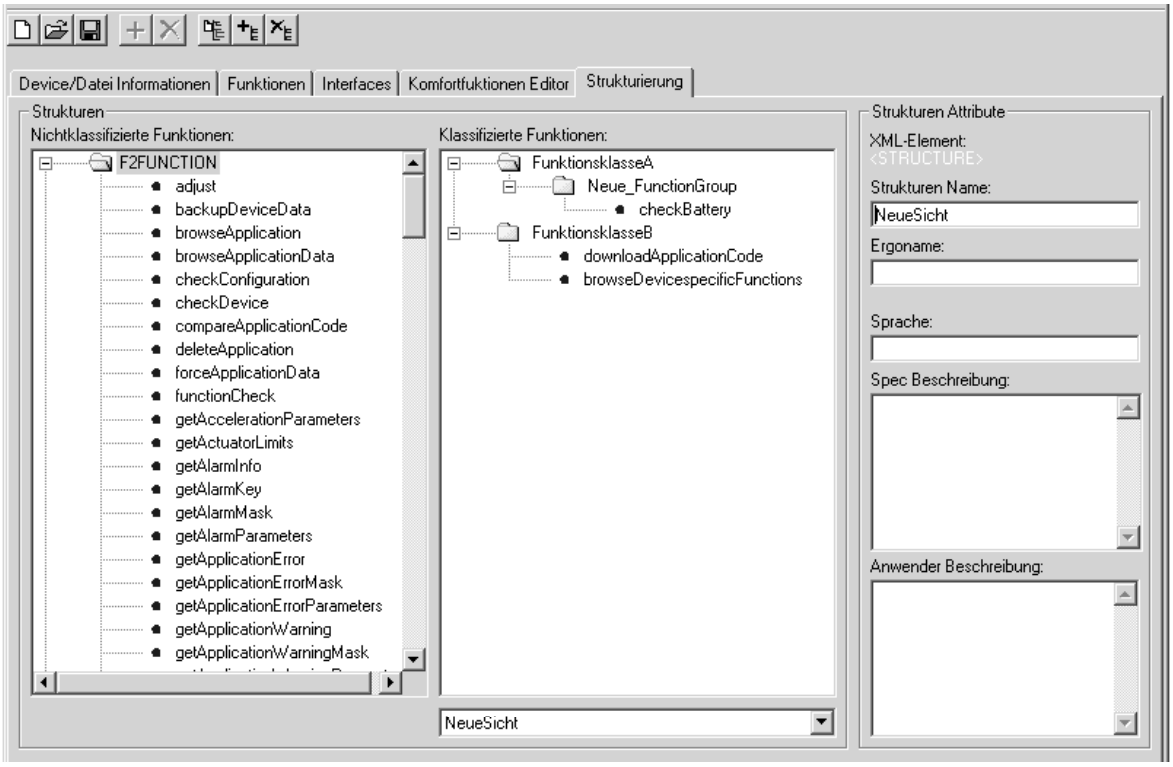


Abbildung 6-3: Erstellung zusätzlicher Strukturierungssichten

Die Karteikarte des Tools zur Erstellung weiterer Klassifikationen ist Abbildung 6-3 zu entnehmen. Im rechten Fensterbereich kann eine neue Strukturierungssicht erstellt werden. Bei der Wahl der Namen für die Sicht und der Funktionsgruppen hat der Benutzer freie Hand. Die Erstellung beliebiger Klassen und Unterklassen ist durchführbar. Anschließend werden die Hilfsfunktionen vom linken Fensterbereich mit dem Mauszeiger in die entsprechende zuvor erstellte Klasse gezogen. Die Funktionen im linken Fensterbereich sind alphabetisch nach Atomar- und Komfortfunktionen aufgelistet.

6.1.3 Evaluierung durch das Projektkonsortium

Die Praxistauglichkeit des realisierten Softwaretools wurde im Rahmen des Verbundprojektes evaluiert. Eine repräsentative Personengruppe aus Geräteherstellern, Anlagenbauern und Anwenderfirmen wurde zum Test herangezogen. Hierzu wurde den Personen die Software in einer Präsentation kurz vorgestellt. Anschließend mussten Geräte- und Systembeschreibungen mit vorgegebenen Eigenschaften erstellt werden. Abschließend folgte mit jedem Probanden ein Gespräch, bei dem ein zuvor erstellter Fragebogen auszufüllen war.

Anschließend erfolgte die Auswertung der Fragebögen. Diese ergab, dass der erforderliche Engineering-Prozess zur Erstellung konformen Beschreibungen

mit dem modellbasierten Verfahren komfortabel und auch für Nichtfachleute in angemessener Zeit durchführbar ist. Die Implementierung der lauffähigen Bedienkomponenten war nicht Bestandteil der Evaluierung. Diese wird durch die Software-Architektur mit wieder verwendbaren Komponenten und der Möglichkeit zur Generierung des Funktionsgerüstes wesentlich vereinfacht, muss jedoch von Programmierern durchgeführt werden (s. Kapitel 6.2, 6.3.3).

6.2 Grundlagen zur Software-Architektur

6.2.1 Proxy-Technologie auf Basis von FDT

Wie bereits in Kapitel 3.3.4 beschrieben spezifiziert FDT eine modulare Software-Architektur. Jedes Softwaremodul, das in FDT integriert werden soll, muss vorgegebene Interfaces implementieren. Die Architektur sieht dabei folgende Bestandteile vor:

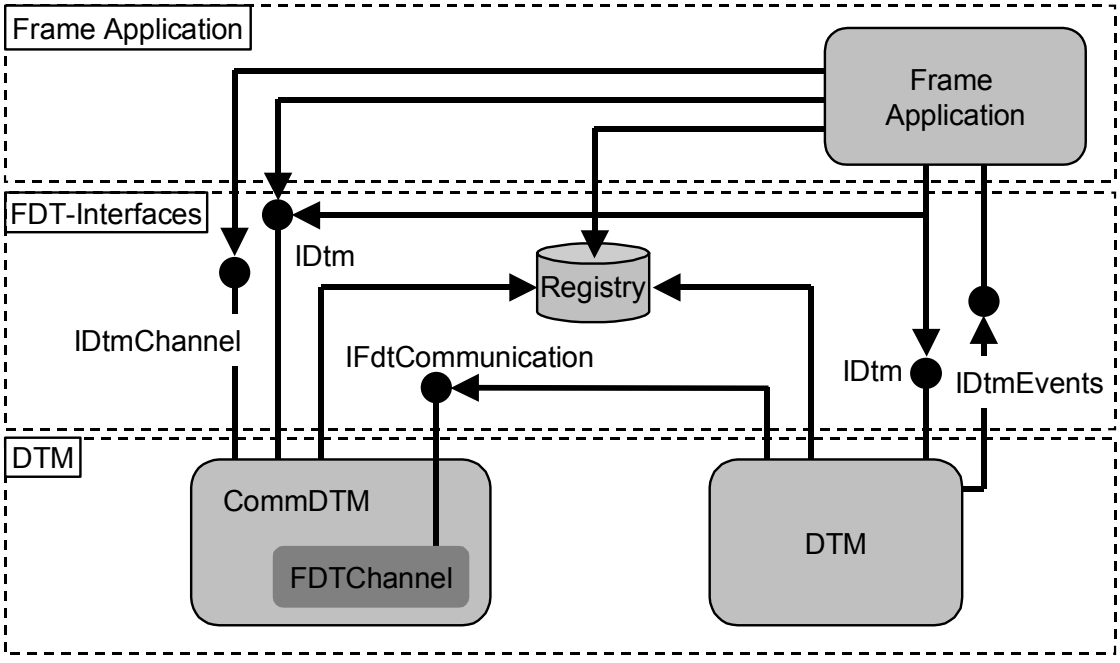


Abbildung 6-4: FDT-Architektur

Die Frame Application, der DTM und der Communication DTM (CommDTM) (s. Abbildung 6-4) bilden die Architektur von FDT. Die Frame Application verwaltet sowohl DTMs als auch CommDTMs und ihre Zuordnung zueinander. Ein DTM ist dabei immer genau einem CommDTM zugeordnet, der für ihn die Kommunikation zum realen Gerät abwickelt. Damit DTMs und CommDTMs in die Frame Application integriert werden können, müssen diese von der Frame

Application erkannt werden. Dazu werden die DTMs mit spezifizierten, so genannten Category IDs[83] in der Windows Registry eingetragen. Die Frame Application kann nun die Windows Registry nach den FDT Category IDs scannen und findet somit alle DTMs und CommDTMs, die für eine Integration in Frage kommen. Folgende Beschreibung der Bestandteile ist in Anlehnung an [PNO 01B] und geht an einigen Stellen auf die DeKOS spezifischen Details der Implementierung ein.

Frame Application

Die Frame Application bildet das Grundgerüst der FDT Architektur. Ihre Aufgabe ist es, die DTMs und CommDTMs verschiedenster Hersteller zu verwalten und zu einem Gesamtsystem zu integrieren. Dazu erstellt die Frame Application eine Bibliothek vorhandener DTMs und CommDTMs, die aufgrund der bereits erwähnten Category IDs in der Windows Registry gefunden werden. Während der Integration, das heißt, während des Zusammenfügens von CommDTM und DTM, stellt die Frame Application sicher, dass ausschließlich Komponenten miteinander verbunden werden, die dasselbe Kommunikationsprotokoll „sprechen“, um spätere Konflikte zu vermeiden.

Device Type Manager

Der Device Type Manager ist in der FDT-Welt der Stellvertreter (Proxy) eines Feldgerätes. Er beinhaltet das Know-How des Herstellers und ermöglicht das Bedienen des Feldgerätes über eine eigene Benutzerschnittstelle. Um in eine Frame Application integriert werden zu können muss der DTM eine Reihe von FDT Interfaces implementieren.

Communication Device Type Manager

Der CommDTM repräsentiert ein bestimmtes Kommunikationsprotokoll. Da ein DTM nicht direkt auf „sein“ Feldgerät zugreifen kann, ist er einem CommDTM zugeordnet. Beliebig vielen DTMs ist die Nutzung ein und des selben CommDTMs möglich. Dieser nimmt die Lese- oder Schreibanfragen von DTMs entgegen und wickelt diese gemäß seines Protokolls ab. Das Ergebnis der Lese- oder Schreibanfrage wird dann an den DTM zurückgeliefert, sodass dieser entsprechend darauf reagieren kann. Um diesen Ablauf zu garantieren muss ein CommDTM spezielle FDT Interfaces implementieren. Da der CommDTM, wie bereits erwähnt, ebenfalls ein DTM ist, müssen die bereits zuvor erwähnten FDT

[83] Eindeutiger Wert (GUID), der eine bestimmte Anwendungsklasse (z. B. OPC-Server) in der Windows Registry kennzeichnet.

Interfaces implementiert sein. Das Kommunikationsprotokoll selbst wird dabei in einem eigenen Objekt, dem so bezeichneten Channel implementiert. So würde zum Beispiel ein Profibus DP CommDTM die Kommunikation über Profibus DP abwickeln.

Integration von FDT Komponenten

Die Integration einer FDT Komponente in die Frame Application erfolgt in zwei Schritten. Im ersten Schritt muss dafür gesorgt werden, dass der DTM von der Frame Application als solcher erkannt wird. Da FDT auf COM basiert ist es zunächst unabdingbare Voraussetzung, dass der DTM als COM Server in der Windows Registry eingetragen ist. Zusätzlich müssen unter diesem Eintrag die bereits erwähnten Category IDs angehängt werden. Diese ermöglichen es der Frame Application, den DTM in der Windows Registry zu finden und anschließend zu instanzieren. Wurde der DTM von der Frame Application erkannt, so steht er zur Integration in ein Projekt zur Verfügung (z. B. in einem Gerätekatalog). Wird nun ein DTM in ein Projekt integriert, so beginnt der zweite Schritt: Der DTM wird initialisiert. Dafür werden Funktionen aus dem Interface `IDtm` in folgender Reihenfolge aufgerufen:

`Environment()`: Der DTM bekommt Informationen über seine Umgebung (Frame Application).

`InitNew()`: Der DTM soll sich initialisieren (z. B. zur Laufzeit benötigte Objekte erzeugen).

`Config()`: Der DTM bekommt Information über die Rolle des Benutzers und seine Rechte.

`SetCommunication()`: Der DTM bekommt einen Zeiger auf das Communication Interface seines CommDTMs.

Wurden all diese Funktionen erfolgreich ausgeführt, so ist der DTM „einsatzbereit“.

6.2.2 Architektur der DeKOS-DTMs

Die grundlegende Architektur eines DeKOS-konformen DTMs ist nachfolgender Abbildung 6-5 zu entnehmen:

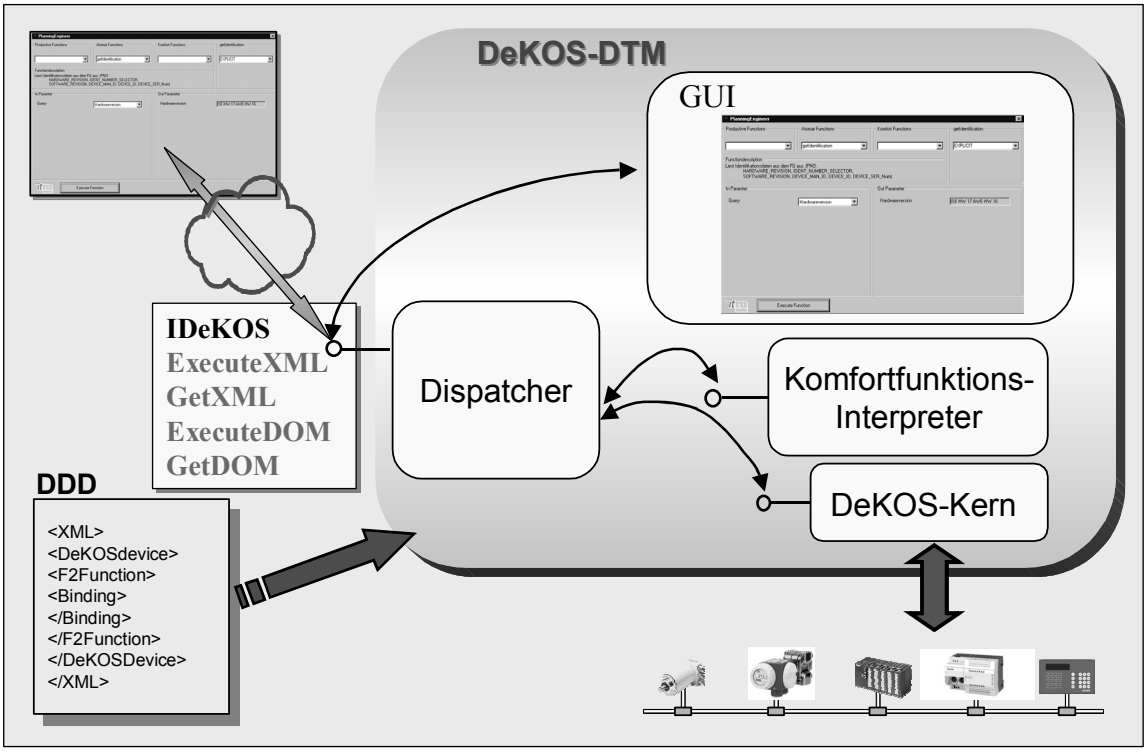


Abbildung 6-5: Software-Architektur der DeKOS-DTMs

Der DTM besteht aus mehreren Komponenten. Als Benutzerschnittstelle dient ein GUI, welches als eigenständige Komponente sowohl innerhalb als auch außerhalb des DTM-Kontexts lauffähig ist. Des Weiteren beinhaltet der DTM diverse ausführende Komponenten wie Dispatcher, Komfortfunktionsinterpreter und DeKOS-Kern. Die Oberfläche des GUI generiert sich dabei vollkommen selbstständig und geräte-/systemunabhängig aus einer entsprechenden DDD, die bei Programmstart eingelesen wird. Das Look&Feel ist hierdurch immer identisch. Von wesentlicher Bedeutung ist der DeKOS-Kern, dessen Funktionsgerüst bei der Implementierung automatisch vom D3Wizard Softwaretool generiert wird. In dieser Komponente erfolgt die Umsetzung der Hilfsfunktionen auf die realen Geräteparameter oder sofern es sich um ein Bedienobjekt eines Partialsystems handelt, der Zugriff auf die entsprechenden unterlagerten Geräte-Proxys. Der Kern stellt die DeKOS-Funktionen über den Dispatcher an der öffentlichen IDeKOS-Schnittstelle zur Verfügung. Die Kommunikation, d. h. die Funktionsaufrufe zwischen den Komponenten erfolgen dabei ausschließlich durch gegenseitigen Austausch von so genannten XML-Aufrufbäumen (s. Abbildung 6-6), in denen die Funktionsaufrufe verpackt sind.

Die beschriebene Architektur des entwickelten DeKOS-DTMs ermöglicht den Zugriff auf die Gerätefunktionalität von anderen Software-Objekten aus. Diese besondere Eigenschaft ist bei der Realisierung der Bedienobjekte für Partialsys-

teme unumgänglich. Die Zugriffsregelung erfolgt gesondert über den DTM-Server (s. Kapitel 6.2.3).

Die DeKOS-Schnittstelle

Über diese Schnittstelle können die Hilfsfunktionen eines Proxys aufgerufen werden. Die Informationen, die für das Ausführen einer Hilfsfunktion benötigt werden sind dabei in einem XML-Dokument, dem Aufrufbaum (s. Abbildung 6-6), enthalten. Dieses XML-Dokument wird beim Aufruf einer DeKOS-Funktion an die Schnittstelle IDeKOS übergeben, und zwar an die Methode `IDeKOS::ExecuteXML()`, wenn das XML-Dokument als ASCII String übergeben wird, und an die Methode `IDeKOS::ExecuteDOM()`, wenn das DOM[84] Objekt des bereits geparsten XML-Dokuments übergeben wird.

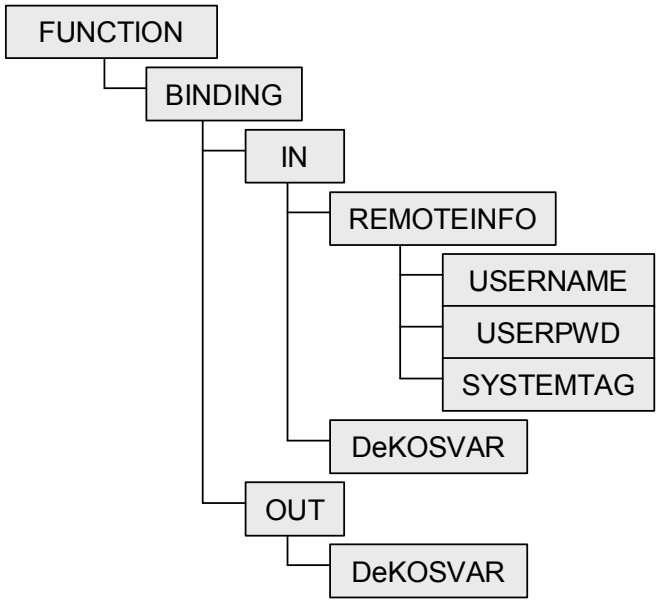


Abbildung 6-6: DeKOS-Aufrufbaum

Der Aufrufbaum enthält alle nötigen Informationen zum Ausführen einer Hilfsfunktion. Er gliedert sich in folgende Teile:

FUNCTION: Das Element enthält Informationen über die aufgerufene DeKOS-Funktion. Die Information wird jeweils in Attributen gespeichert.

[84] Document Object Model

DeKOSNR: Die DeKOSNR ist die eindeutige Identifikation der aufzurufenden DeKOS-Funktion.

NAME: Beinhaltet den eindeutigen Namen der DeKOS-Funktion.

BINDING:

BINDTYPE: Das DeKOS-Bedienmodel definiert mehrere Arten, wie eine Funktion aufgerufen werden kann. Dies wird im Attribut BINDTYPE angegeben.

IN: Das IN Element enthält die Aufrufparameter der Funktion, die in den Elementen DeKOSVAR enthalten sind. Benötigt eine Funktion keine Aufrufparameter (z. B. Reset), so bleibt das Element leer.

REMOTEINFO: Dieses Element beinhaltet alle Informationen, die für einen entfernten Funktionsaufruf nötig sind. Dies sind im einzelnen:

USERNAME: Das Element enthält den Namen eines Benutzers, der Funktionalität entfernt aufrufen möchte. Dies kann sowohl eine Person als auch eine Softwarekomponente wie zum Beispiel ein Subsystem-Proxy sein.

USERPWD: Dieses Element enthält das Passwort des betreffenden Benutzers. Damit wird verhindert, dass Funktionalität unbefugt aufgerufen wird.

DEVICETAG: Das Element enthält einen eindeutigen Identifier für die Softwarekomponente, an der die Funktionalität aufgerufen werden soll.

OUT: Im Element OUT werden die Rückgabewerte der Funktion abgelegt. Die einzelnen Werte werden dabei analog zum Element IN in den Elementen DeKOSVAR gespeichert.

DeKOSVAR: Das Element repräsentiert einen beliebigen Wert und findet Anwendung um die Aufruf- bzw. Rückgabewerte einer Funktion zu speichern. Es beinhaltet folgende Attribute:

NAME: Der Name der Variable (z. B. „Password“ wenn ein Passwort innerhalb der Funktion „login“ übergeben werden soll).

VALUE: Der eigentliche Wert der Variable (z. B. „SomePassword“).

TYPE: Der Datentyp der Variable (z. B. „STR“ um anzuzeigen, dass die Variable vom Datentyp String ist).

Durch den Aufrufbaum ist es möglich, eine tatsächliche Trennung zwischen dem Aufrufer einer Funktion und dem Erbringer der Funktionalität durchzuführen. Konkret bedeutet dies, dass ein Partialsystem-Proxy auf die Funktionalität seiner unterlagerten Geräte-Proxys zugreifen kann, ohne dass Partialsystem - Proxy und Geräte-Proxy in ein und derselben Softwarekomponente enthalten sein müssen.

6.2.3 DTM-Server

Im Rahmen der Anforderungsanalyse wurden für das Konzept einer universellen Kommunikationsschnittstelle folgende Anforderungen identifiziert:

Sicherheit

Da das Konzept einen Fernzugriff auf Gerätefunktionalität durch das Bedienobjekt des Partialsystems ermöglichen soll, steht das Thema Sicherheit bei den Anforderungen an erster Stelle. Im Einzelnen sind folgende Punkte zu berücksichtigen:

- *Schutz vor unbefugtem Zugriff*: Es darf nur mit entsprechenden Zugriffsrechten möglich sein, auf Gerätefunktionalität zuzugreifen. Dies schließt die Authentifizierung des Benutzers ein.
- *Differenzierter Zugriff*: Ein Benutzer darf nicht automatisch Zugriff auf alle Bedien-Proxys haben. Zum Beispiel soll ein Hersteller nur auf seine Geräte innerhalb der Anlage zugreifen können.
- *Sicherer Datenaustausch*: Da die Kommunikation auch über Internet erfolgen kann, muss sichergestellt werden, dass die übermittelten Daten für Dritte nicht lesbar sind.

Zugriffsmanagement

Ein entsprechendes Zugriffsmanagement soll den Zugriff auf Geräte-Proxys und damit auf die Feldgeräte regeln. Dabei ist zu beachten:

- *Mehrfachzugriffe verhindern*: Es muss verhindert werden, dass mehrere Benutzer/Objekte gleichzeitig auf dasselbe Geräte-Proxy zugreifen.
- *Priorität lokaler Benutzer*: Es muss die Möglichkeit gegeben sein, einem lokalen Benutzer den Vorrang zu geben. Insbesondere darf es nicht sein, dass ein entfernter Benutzer die lokale Benutzung blockiert. Dies ist vor allem im Falle einer Störung der Anlage wichtig.

Konzept

Ausgehend von den in der Anforderungsanalyse festgelegten Punkten wurde im Rahmen einer Semesterarbeit [GROSSMANN 02] ein Konzept entwickelt, das diesem in vollem Umfang gerecht wird. Kernstück des Konzeptes ist eine Server-Komponente, der so genannte DTM-Server. Dieser DTM-Server soll den Zugriff auf die Geräte-Proxys durch ein DTM auf Partialsystemebene ermögli-

chen. Aus diesem Grunde ist der DTM-Server selbst wiederum ein DTM, der in die Frame Application integriert wird. Dadurch kann der DTM-Server auf die FDT Interfaces der Frame Application zugreifen (Abbildung 6-7). Dies ermöglicht dem DTM-Server den Zugriff auf Informationen über das Gesamtsystem (z. B. welche DTMs verfügbar sind) sowie auf andere DTMs.

Empfängt der DTM-Server nun einen Aufruf, so kann er ihn an den entsprechenden DTM weiterleiten. Welche DTMs in der Frame Application verfügbar sind, kann der DTM-Server von der Frame Application abfragen. Damit ist der DTM-Server die zentrale Komponente für den Zugriff auf die Bedien-Proxys. Dies stellt sicher, dass ein Zugriff auf lokale Bedien-Proxys nur über das Sicherheitskonzept des DTM-Servers möglich ist, das nicht umgangen werden kann. Da das FDT-Konzept selbst auf COM basiert, kommt auf Betriebssystemebene das COM-Sicherheitskonzept sowie Benutzerverwaltung zum Tragen. Eventuelle Sicherheitslücken von COM können durch das Konzept nicht behoben werden. Der Zugriff auf den DTM-Server wiederum kann über COM oder DCOM erfolgen. Dies deckt den Fall für einen Zugriff innerhalb eines Intranets ab. Soll der DTM-Server über Internet, also weltweit zugreifbar sein, so kommen zusätzlich Webtechnologien ins Spiel.

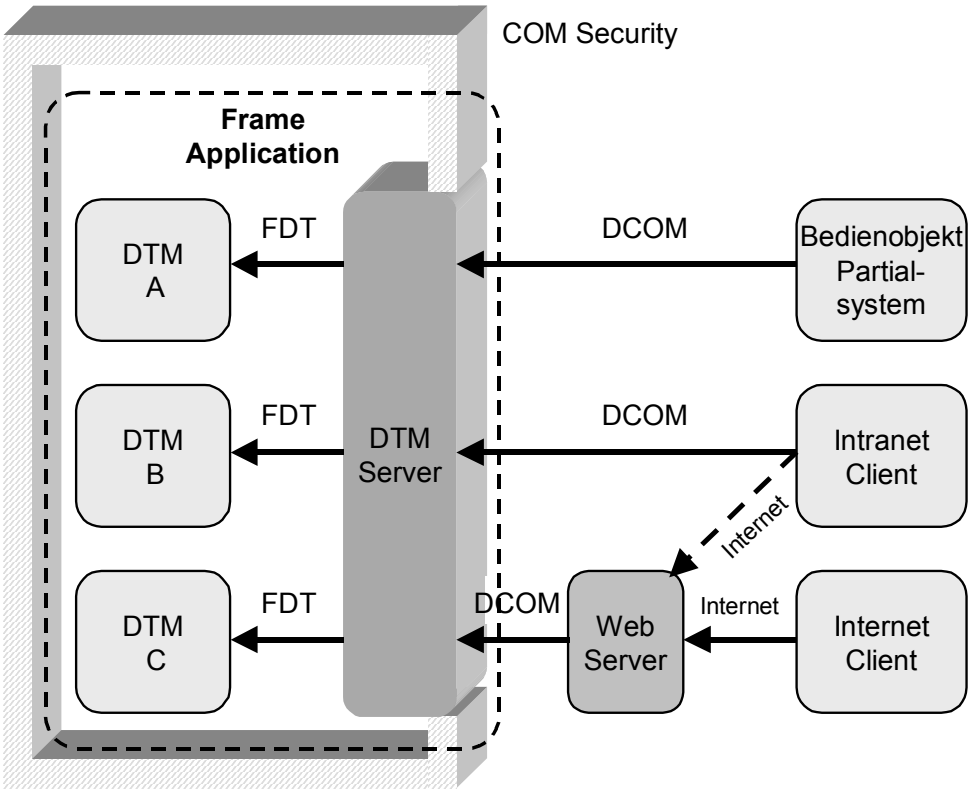


Abbildung 6-7: DTM-Server Konzept [GROSSMANN 02]

Architektur des DTM-Servers

Der DTM-Server ist das Kernstück der Architektur (Abbildung 6-8) und ist selbst als DTM konzipiert. Damit hat er Zugriff auf die FDT-Interfaces der Frame Application. Dies wiederum ermöglicht dem DTM-Server den Zugriff auf die DTMs des Gesamtsystems. Welche Funktionen dabei von Bedeutung sind, wird an späterer Stelle noch im Detail behandelt. Der DTM-Server selbst setzt sich aus drei wesentlichen Komponenten zusammen. Dies sind *FDT Components*, *Users* und *Remote Access*:

- *FDT Components*: Diese Komponente verwaltet die verfügbaren DTMs des Systems. Sie ermittelt, welche DTMs verfügbar sind und regelt den Zugriff auf diese. Zusätzlich können hier Zugriffsrechte für die einzelnen DTMs vergeben werden.
- *Users*: Dieses Modul übernimmt das User Handling. Hier werden die User und ihre Rechte verwaltet. Möchte ein User auf einen Bedien-Proxy zugreifen, so wird hier überprüft, ob er dazu befugt ist.
- *Remote Access*: Dieses Modul implementiert das `IDeKOS` Interface und ist der Einstiegspunkt für entfernte Aufrufe. Es empfängt die Aufrufbäume und leitet sie an den entsprechenden DTM weiter, nachdem die Rechte des Aufrufers verifiziert wurden. Dies geschieht durch das Modul Users. Um den Aufrufbaum weiterleiten zu können wird der entsprechende Zeiger auf das Interface `IDeKOS` des adressierten DTMs vom Modul *FDT Components* erfragt.

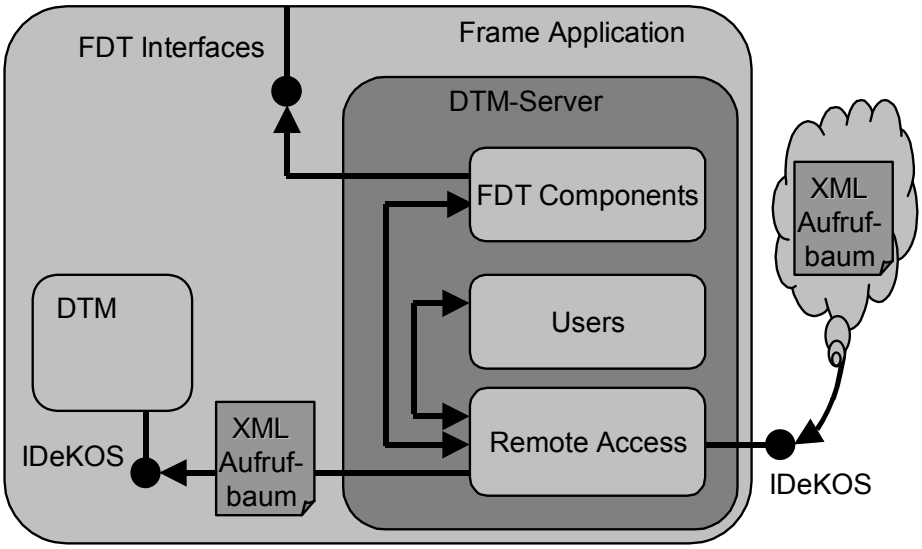


Abbildung 6-8: DTM-Server Architektur [GROSSMANN 02]

Der Betrieb des DTM-Servers gliedert sich nach folgendem Ablauf:

Zu Beginn greift der DTM-Server auf die Interfaces der Frame Application zu, um eine Liste der vorhandenen DTMs zu erhalten. Nach diesem „Scan“ kennt der DTM-Server alle DTMs innerhalb der Frame Application. Empfängt der DTM-Server nun einen Aufrufbaum über die `IDeKOS` Schnittstelle am Modul *Remote Access*, so werden zunächst die Rechte des Aufrufers durch das Modul *Users* verifiziert. Sind die Zugriffsrechte des Users ausreichend, wird ein Zeiger auf die `IDeKOS` Schnittstelle des adressierten DTMs über das Modul *FDT Components* erfragt. Nun kann der Aufrufbaum an den DTM weitergeleitet werden. Dieser führt die angegebene Hilfsfunktion aus und liefert das Ergebnis an den DTM-Server zurück. Dort wird der Aufrufbaum, der das Ergebnis enthält, an den Aufrufer zurückgeschickt. Wird auf den DTM lokal zugegriffen, so wird dies durch einen Remote Zugriff nicht gestört. In diesem Fall wird eine entsprechende Meldung an den entfernten Aufrufer gesendet.

6.3 Beispiel Pumpstation

Zur anschaulichen Erläuterung der einzelnen Schritte des notwendigen Engineerings wurde ein Beispiel aus dem Bereich der Verfahrenstechnik gewählt. Es handelt sich hierbei um eine Pumpstation. Diese besteht aus einer Pumpe, einem Stellventil und einem Durchflussmesser, die bezüglich des strömenden Fluids in Reihe geschaltet sind. Die Steuerintelligenz wird von einer Siemens S7-400 erbracht. Durch diese Kombination ist es möglich, definierte Mengen weiter zu pumpen bzw. das gepumpte Volumen zu messen.

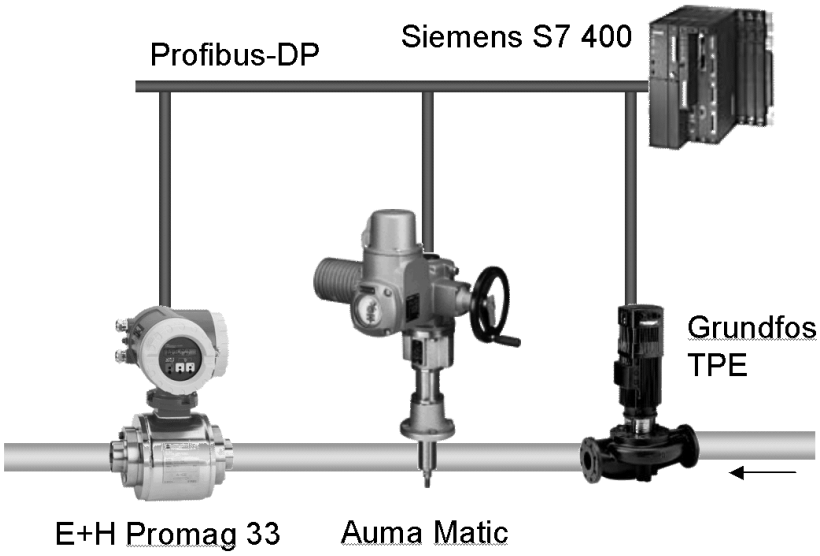


Abbildung 6-9: Gerätetechnischer Aufbau der Pumpstation

Bei dem ersten Feldgerät handelt es sich um eine kompakte Kreiselpumpe aus der Reihe TPE mit dem Profibus-DP Modul G100 der Firma Grundfos. Sie erreicht einen maximalen Betriebsdruck von 1 MPa. Das nächste Gerät ist ein Stellantrieb der Firma Auma, bestehend aus einem Drehantrieb und der Steuerung Auma Matic mit Profibus-DP Kommunikationsmodul. Der Drehantrieb ist mit einem Stellventil verbunden. Als drittes Gerät ist der Promag 33, ein magnetisch-induktives Durchfluss-Messgerät der Firma Endress+Hauser, eingebaut. Die Messung basiert auf dem Faradayschen Induktionsgesetz, welches besagt, dass in einem Leiter, der sich in einem Magnetfeld bewegt, eine Spannung induziert wird. Das fließende Medium ist in diesem Fall der bewegte Leiter, dessen induzierte Spannung gemessen und dann über den Rohrquerschnitt der Gesamtdurchfluss berechnet wird. Das Gerät besitzt ebenfalls eine Profibus-DP Kommunikationsanschaltung.

Die Pumpstation hat folgende Prozessfunktionalität:

1. Zuvor festgelegtes Volumen durchpumpen.
2. Pumpen bis ein bestimmtes Ereignis eintritt und Meldung des gepumpten Volumens rückmelden.

6.3.1 Erstellung der Gerätebeschreibungen

Da für die Geräte der Pumpstation keine DeKOS-konformen Beschreibungen vorliegen, werden diese zuvor erstellt. Das Ziel ist die Generierung eines Bedienobjektes für das Partialsystem. Dementsprechend ist es obligatorisch, dass die unterlagerten Feldgeräte die Systemgrundfunktionen unterstützen müssen.

Nachfolgende Tabellen geben Aufschluss über die Hilfsfunktionalität der Geräte. Alle verwendeten Feldgeräte haben die Gerätegrundfunktionen in der Firmware implementiert und unterstützen die Systemgrundfunktionen. Da es sich bei der Erstellung der Gerätebeschreibungen nicht um eine Neuentwicklung der Geräte-Firmware handelt, sondern im Rahmen einer Migrationsstrategie existierende Feldgeräte Verwendung finden sollen, muss zuvor natürlich sichergestellt sein, dass die Geräte die geforderte Funktionalität erbringen können. Die Abarbeitung der Hilfsfunktionen im Gerät durch Schreib-/Lesezugriffe der entsprechenden Parameter ist den jeweiligen Benutzerhandbüchern zu entnehmen und erst für die Phase der Implementierung relevant. Zusätzlich weisen die Geräte noch nachfolgende Prozess- und Hilfsfunktionen auf, auf deren Beschreibung verzichtet wird (s. [BENDER ET AL. 02]):

Prozessfunktion
startPump
Hilfsfunktionen
getDevicespecificParameters, setDevicespecificParameters

Tabelle 6-1: Zusätzliche Gerätefunktionen der Pumpe

Bei der intelligenten Pumpe handelt es sich um ein vergleichsweise einfaches Gerät. Hieraus resultiert, dass neben den Gerätegrundfunktionen nur wenige Hilfsfunktionen unterstützt werden (s. Tabelle 6-1).

Prozessfunktion
openValve
Hilfsfunktionen
browseApplicationData, getAccelerationParameters, setAccelerationParameters, getConfiguration, setConfiguration, getControlMode, setControlMode, getControlParameters, setControlParameters, getDevicespecificParameters, setDevicespecificParameters, getFixedValues, setFixedValues, getMeasurementResolution, setMeasurementResolution, getRedundancyConfiguration, setRedundancyConfiguration, getRedundancyParameters, setRedundancyParameters

Tabelle 6-2: Zusätzliche Gerätefunktionen des Stellantriebs

Prozessfunktion
getFlowRate
Hilfsfunktionen
browseApplicationData, forceApplication, resetApplication, startCalibration, uploadApplicationCode, getCalibrationParameters, setCalibrationParameters, getCertification, getConfiguration, getControlParameters, setControlParameters, getDevicespecificParameters, setDevicespecificParameters, getDiagMode, setDiagMode, getDiagnosis, getDisplayParameters, setDisplayParameters, getEventReaction, setEventReaction, getFilteringParameters, setFilteringParameters, getFixedValues, setFixedValues, getMaxMeasurementValue, setMaxMeasurementValue, getMeasurementDimension, setMeasurementDimension, getMeasurementOffsetValue, setMeasurementOffsetValue, getMinMeasurementValue, setMinMeasurementValue, getTagName, setTagName, getUnit, setUnit

Tabelle 6-3: Zusätzliche Gerätefunktionen des Durchflussmessers

Im Vergleich hierzu unterstützt der Stellantrieb und der Durchflussmesser eine große Zahl an Hilfsfunktionen, die bei der Integration bis hin zur Instandhaltung vom Bediener handhabbar sein müssen. An dieser Stelle wird schnell offensichtlich, dass ohne entsprechende Klassifikation der Funktionsbibliothek das Engineering beim Erstellen der Beschreibungen und bei der Nutzung des fertigen Bedienobjektes kaum möglich ist.

Zur Erstellung der Gerätebeschreibungen ist für jedes Feldgerät entsprechend Abbildung 6-10 vorzugehen:

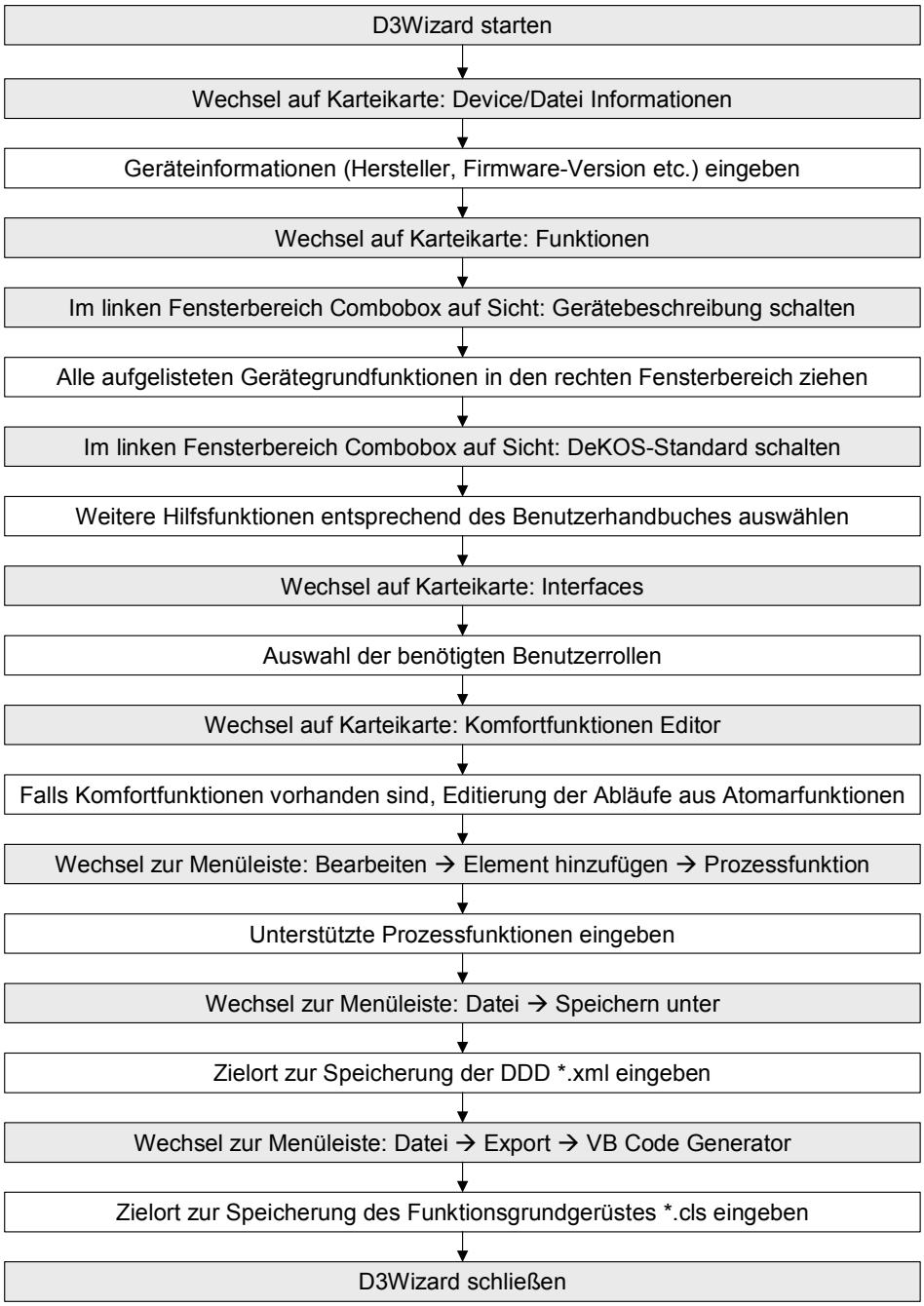


Abbildung 6-10: Vorgehensweise zur Erstellung einer Gerätebeschreibung

Auf Basis der konform erstellten Gerätebeschreibungen können unter zu Hilfenahme der Funktionsgerüste in Form der Klassendateien und der wieder verwendbaren Software-Komponenten die Bedienproxys erzeugt werden (s. Kapitel 6.3.3). Auf einen Abdruck der DDDs im Anhang wird aufgrund des Umfangs verzichtet.

6.3.2 Erstellung der Systembeschreibung

Aufbauend auf den zuvor erstellten Gerätebeschreibungen soll im nächsten Schritt die Systembeschreibung der Pumpstation erfolgen. Das Verfahren zur Generierung einheitlicher Bedienobjekte für verteilte mechatronische Systeme sieht vor, dass auf Basis der Gerätegrundfunktionen die Systemgrundfunktionen (s. Tabelle 5-1) ohne zusätzlichen Aufwand durch ihren fraktalen Charakter partialsystemseitig zur Verfügung stehen.

Neben diesen allgemeinen Hilfsfunktionen soll das Partialsystem eine weitere Diagnosefunktion unterstützen. Diese führt eine Art Selbstdiagnose des Systems aufgrund der Systemlogik aus. Der Verschleiß an der Dichtung des Stellventils ist dahingehend zu überprüfen, dass bei geschlossenem Ventil die Pumpe kurz anläuft, um einen ausreichenden Druck in der Rohrleitung aufzubauen. Falls der Durchflussmesser einen Durchfluss registriert, ist die Dichtung des Ventils schadhaft und instand zu setzen. Da der Verschleiß in der Realität kontinuierlich statt findet, und die Selbstdiagnose mit einem höheren Druck arbeitet als der Betriebsdruck, kann ein Schwellwert über die Durchflussmenge eingestellt werden, ab dem die Diagnosefunktion Alarm schlägt. Zusätzlich sind die beschriebenen Prozessfunktionen (s. Kapitel 6.3) in die Beschreibung zu integrieren.

Zur Erstellung der Systembeschreibung für das Partialsystem Pumpstation ist, wie nachfolgend beschrieben, vorzugehen. Das Verfahren gestaltet sich dabei zweiteilig. Im ersten Schritt wird äquivalent zur Erstellung einer Gerätebeschreibung verfahren (s. Abbildung 6-11).

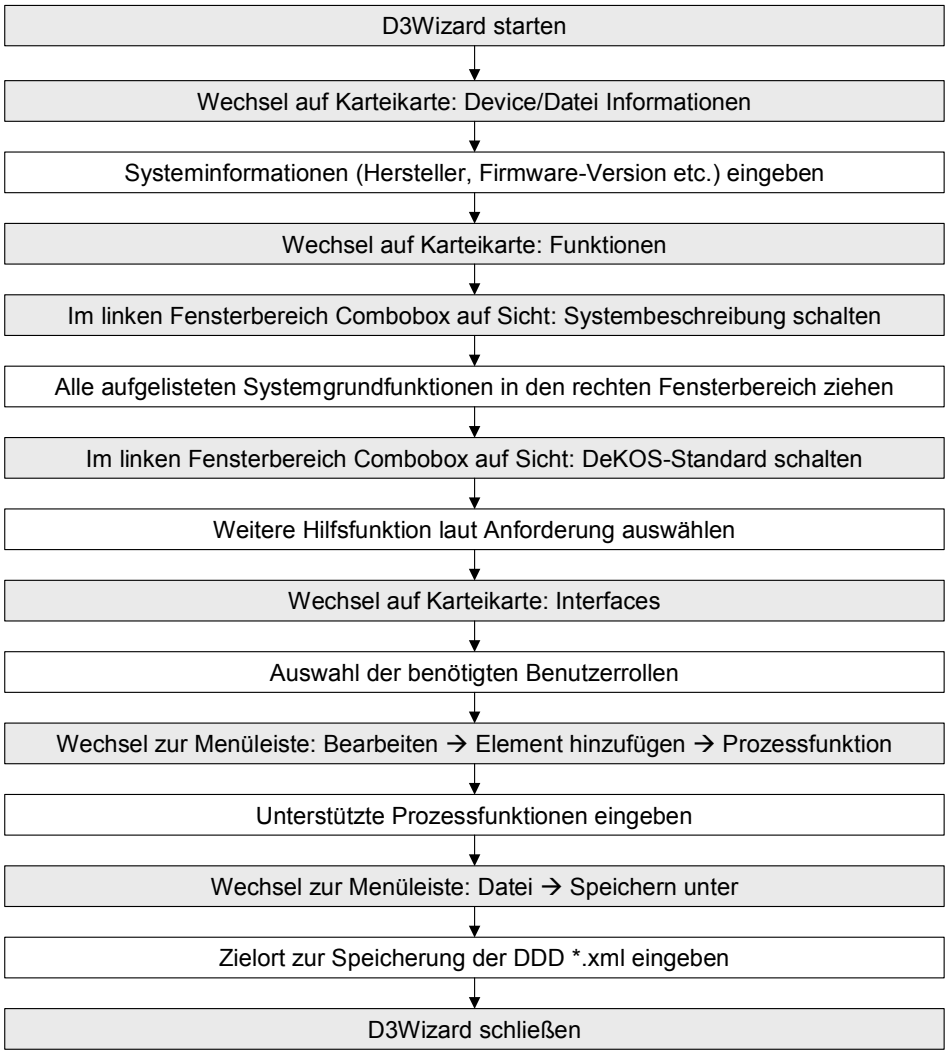


Abbildung 6-11: Vorgehensweise zur Erstellung einer Systembeschreibung

Als Ergebnis des ersten Schrittes liegt das „Rohgerüst“ der Systembeschreibung vor. Diese beinhaltet zwar alle Funktionen, die das Partialsystem zu unterstützen hat, die Verknüpfungen zu den unterlagerten Feldgeräten und der Algorithmus zur Realisierung der polymorphen Hilfsfunktion sind darin aber noch nicht enthalten. Dies geschieht im zweiten Schritt. Hierzu kann nachfolgend unter Verwendung des prototypischen Verknüpfungsgenerators vorgegangen werden (s. Abbildung 6-12):

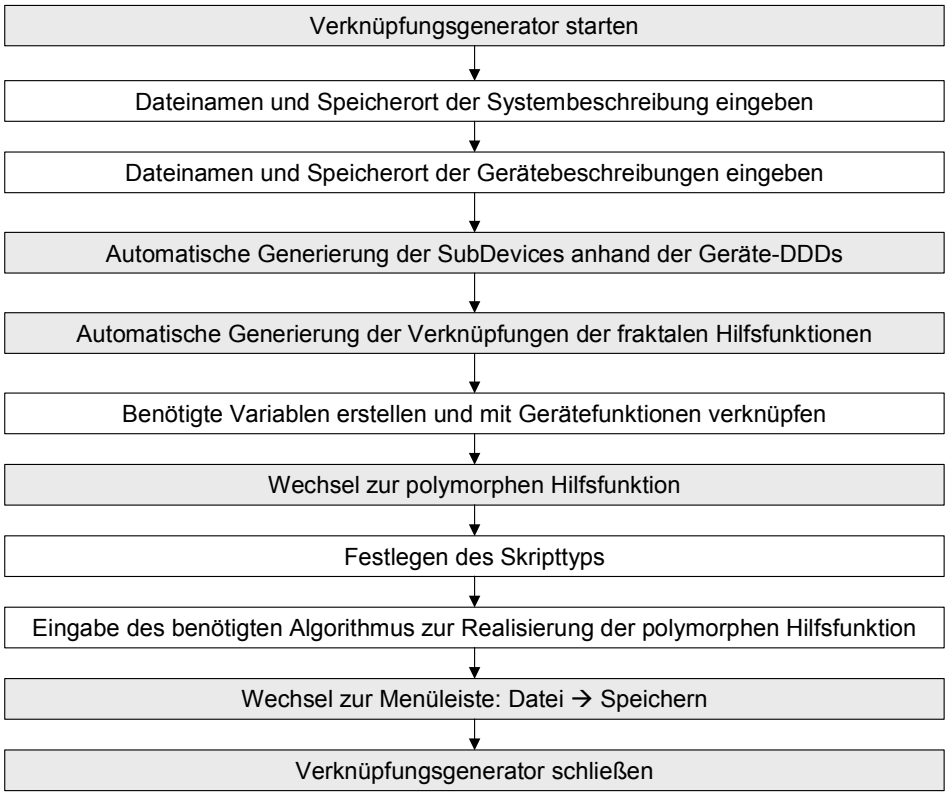


Abbildung 6-12: Verknüpfung der System-DDD mit den Geräte-DDDs

Durch die Eingabe der Dateinamen erkennt das Tool die Systembeschreibung mit den dazu gehörigen Gerätebeschreibungen. Eine automatische Generierung der SubDevice-Informationen ist anhand der Geräte-DDDs gegeben. Optional können die GUIDs vergeben werden. Wie in Kapitel 5.2.2 ausführlich dargelegt, ist für jede fraktale Hilfsfunktion die Abarbeitungslogik eindeutig in der Referenzarchitektur beschrieben. Dementsprechend können die Verknüpfungen, nachdem die unterlagerten Geräte über die SubDevice-Informationen anhand der Device-IDs eindeutig zu referenzieren sind, automatisch erstellt werden. In Kapitel 5.2.3 wurde dargelegt, dass die Erstellung polymorpher Hilfsfunktionen ein Kreativprozess ist und nur mit ausreichendem Systemwissen durchführbar ist. Somit kann die Festlegung der Variablen nur manuell erfolgen. Gleiches gilt für den Algorithmus zur Abarbeitung. Die fertig gestellte DDD des Beispielsystems ist Anhang G zu entnehmen.

Nachdem die vollständige DDD des Partialsystems erstellt wurde, kann hieraus das Funktionsgerüst für die Implementierung generiert werden. Dazu ist der D3Wizard erneut zu starten. Nach einem Einlesen der System-DDD kann über die Exportfunktion → VB Code Generator die Generierung erfolgen.

6.3.3 Halbautomatische Generierung der DTMs

Nach der Fertigstellung der Partialsystem-DDD und der Geräte-DDDs kann mit der Implementierung der Proxys begonnen werden. In Kapitel 6.2.2 wurde die Software-Architektur der DeKOS-DTMs erläutert. Dem Programmierer werden diverse Software-Komponenten zur Verfügung gestellt, die unverändert in den jeweiligen VB-Projekten verwendet werden können. Ausschließlich beim beschriebenen DeKOS-Kern ist Programmierungsaufwand notwendig. Hierzu ist wie beschrieben das Wissen über die Ausführung der Hilfsfunktionen durch reale Geräteparameter erforderlich. Gleiches betrifft das System-Bedienobjekt; die realen Aufrufe der Funktionen aus dem Systemproxy heraus bedürfen einer Ausprogrammierung des zuvor generierten Funktionsgerüstes.

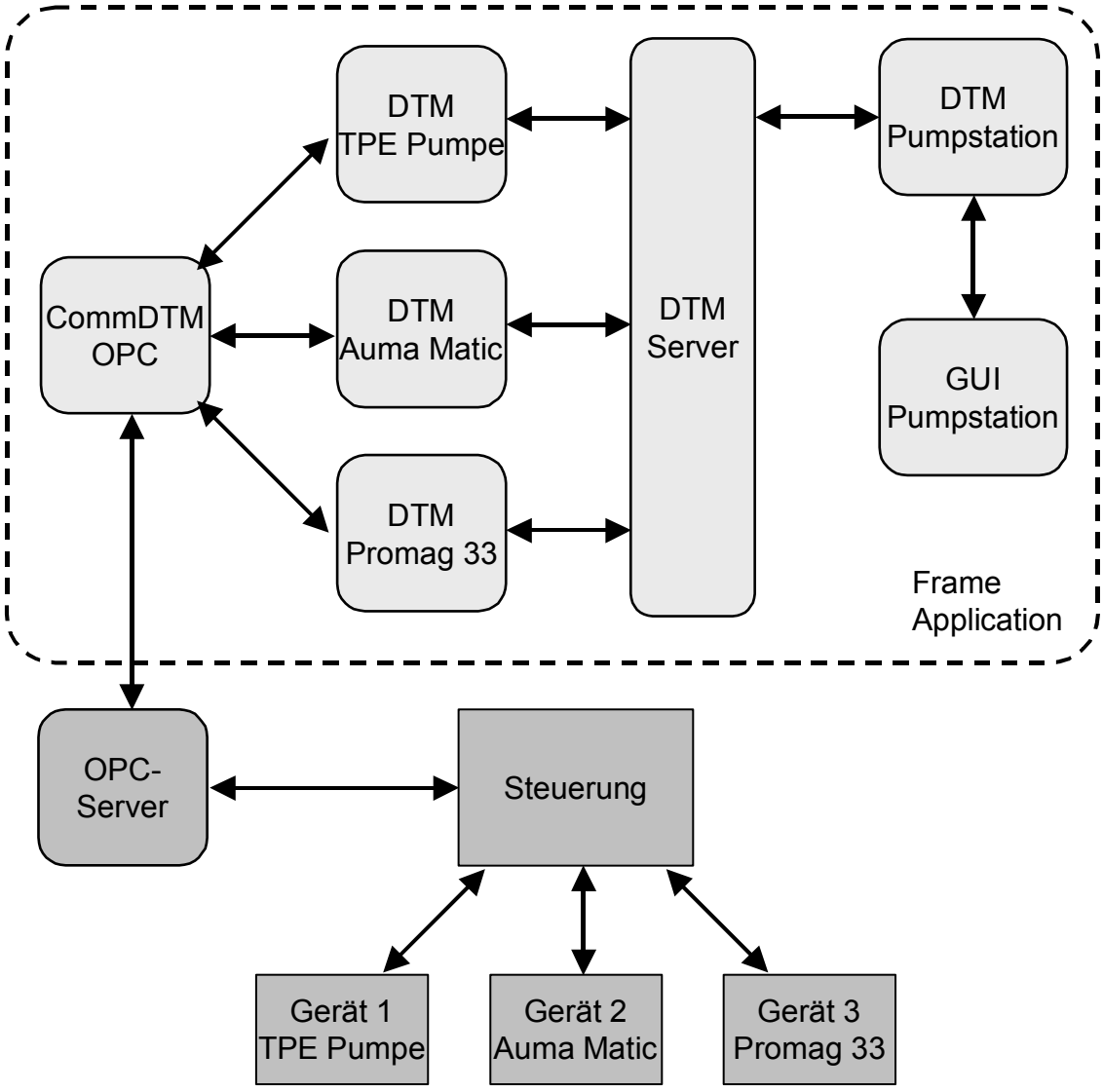


Abbildung 6-13: Gesamtarchitektur zur Bedienung eines Partialsystems

Sind alle DeKOS-Kerne implementiert und anhand der VB-Projekte der lauffähige Code erzeugt, kann mit der Integration der DTMs in die FDT-Umgebung (s. Kapitel 6.2.1) begonnen werden. Die Gesamtarchitektur ist Abbildung 6-13 zu entnehmen. Die Kommunikation zwischen der Frame Application und der Steuerung wurde über eine OPC[85]-Verbindung realisiert. Hierfür sind sowohl kommerzielle Kommunikations-DTMs als auch OPC-Server für verschiedenste Steuerungen erhältlich. Die Geräte sind über den Feldbus Profibus-DP mit der Steuerung verbunden. Sind alle notwendigen Software-Komponenten registriert und die DTMs gestartet, steht dem Bediener das Bedienobjekt des Partialsystems Pumpstation auf der Engineering-Workstation zur Verfügung (s. Abbildung 6-14).

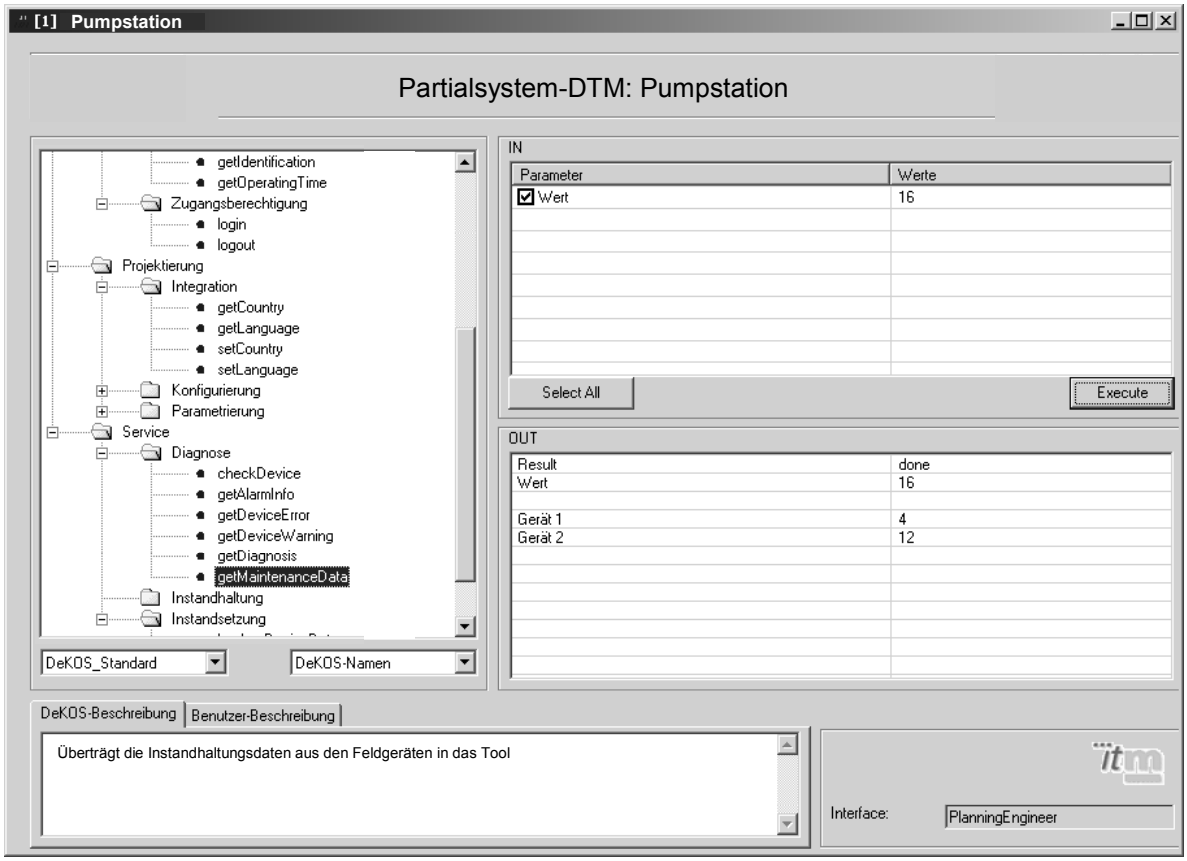


Abbildung 6-14: Bedienobjekt des Partialsystems Pumpstation

Wie in der Abbildung 6-14 zu sehen ist, werden die Hilfsfunktionen entsprechend der spezifizierten Strukturierungssichten angeboten. Auch im GUI ist ein Umschalten der Sichten über die entsprechende Combobox jederzeit möglich.

[85] Ole for Process Control: COM-basierende Kommunikationstechnologie im Bereich der Automatisierungstechnik.

Zunächst wird die fraktale Hilfsfunktion `getMaintenanceData` vom GUI des Partialsystems aufgerufen. Der DeKOS-Kern liefert daraufhin die Parameterliste. Der Bediener kann nun den Wert 16 für die Anfrage, ob innerhalb der nächsten 16 Stunden Wartungsarbeiten anfallen, im oberen Fensterbereich IN eintragen und durch Betätigen des Execute-Buttons ausführen. Im Kern werden die Aufrufbäume für die unterlagerten Geräte-Proxys generiert und an die entsprechenden DTMs gesendet. Diese verarbeiten die Aufrufbäume und liefern die Ergebnisse an den Partialsystem-Kern zurück, der nach der Auswertung das Resultat an das GUI sendet und im Bereich OUT darstellt.

6.4 Zusammenfassung

Einleitend wurde der D3Wizard, ein Tool mit graphischer Benutzeroberfläche zur Erstellung DeKOS-konformer Beschreibungen, vorgestellt. Die komfortablen Möglichkeiten zur Tool-gestützen Generierung einer DDD und zur Erstellung zusätzlicher Strukturierungssichten wurden beschrieben.

Um die Realisierung der lauffähigen Proxys zur Bedienung eines Partialsystems nachvollziehen zu können, folgte eine Einführung in die Grundlagen der eingesetzten Technologien. Insbesondere wurde die Gesamtarchitekur der verwendeten Proxy-Technologie FDT beschrieben. Die notwendigen Modifikationen an den DTMs zur Nutzung der Funktionalität durch andere Software-Objekte wurde dargelegt. Erklärungen zum DTM-Server zur Koordination der verschiedenen Proxys, Regelung der Zugriffsrechte etc. folgten.

Das abschließende Beispiel aus der Verfahrenstechnik gab die notwendigen Schritte zur Erstellung eines Bedienobjektes auf Partialsystemebene wieder. Die Pumpstation sollte zwei Prozessfunktionen unterstützen und hinsichtlich der Hilfsfunktionalität neben den Systemgrundfunktionen eine weitere polymorphe Hilfsfunktion zur speziellen Systemdiagnose unterstützen. Die hierzu erforderlichen Schritte zur Erstellung der Gerätebeschreibungen, der Systembeschreibung und der Verknüpfung folgten. Zum Schluss wurde kurz auf die Implementierung der Proxys eingegangen und das lauffähige Bedienobjekt vorgestellt.

Der Arbeitsaufwand zur modellbasierten Generierung eines entsprechenden Bedienobjektes auf Partialsystemebene scheint auf den ersten Blick sehr hoch. Allerdings sollte berücksichtigt werden, dass im beschriebenen Beispiel sowohl konforme Gerätebeschreibungen zu erstellen als auch die DTMs zu implementieren waren. Beschränken sich die notwendigen Maßnahmen ausschließlich auf die Erstellung der Systembeschreibung und des Proxys für das Partialsystem, ist es durch das Modell und das toolgestützte Verfahren mit geringem Aufwand möglich, ein Bedienobjekt zu generieren.

7 Abschlussbemerkungen

Inhalt des abschließenden Kapitels ist die Zusammenfassung der vorliegenden Arbeit. Erläuterungen zu den wichtigsten Aspekten des Lösungskonzeptes und eine Bewertung der vorgestellten Lösung anhand der prototypischen Umsetzung, dargelegt in Kapitel 6, folgen. Ein Ausblick auf nachfolgende Arbeiten wird gegeben.

7.1 Zusammenfassung

Die kontinuierliche Forderung der Betreiber nach Steigerung der Produktivität, Qualität und Flexibilität ihrer automatisierungstechnischen Anlagen hat zu erheblichen Veränderungen seitens des gesamten Engineering-Prozesses geführt. Die entstandenen zusätzlichen Kosten betreffen sowohl die Investitionen als auch den laufenden Betrieb der komplexen Anlagen. Der Wandel betrifft primär den Anlagenbetreiber, der in der Lage sein muss, die Komplexität seiner Systeme bei der täglichen Arbeit noch beherrschen zu können. Sekundär ist aber auch der Gerätehersteller und Anlagenbauer bedingt durch die Anwenderanforderungen betroffen.

Die Synergie aus einer komponentenbasierten Sicht und dem Bestreben nach Standardisierung der Hilfsfunktionalität führt zum Ansatz der einheitlichen Bedienung auf Partialsystemebene. Dieses Konzept sieht eine Kapselung der Hilfsfunktionalität von Anlagenteilen vor. Dementsprechend wird dem Bediener die vorhandene Funktionalität zur Wartung, Diagnose etc. eines Partialsystems über standardisierte Hilfsfunktionen angeboten. Die Geräteabhängigkeiten, das notwendige Wissen der speziellen Systemeigenschaften und die erforderliche Bedienlogik bleiben dem Bediener verborgen. Diese sind im Bedienobjekt des Partialsystems realisiert. Zusätzliche Diagnosemöglichkeiten auf Partialsystemebene, wie im Rahmen des Asset Managements gefordert, sind berücksichtigt.

Die Verwirklichung einer derartigen Lösung muss dabei aufwandsarm durch den Anlagenbauer umsetzbar sein. Dieser ist wiederum auf die Gerätehersteller angewiesen, die Feldgeräte zur einfachen Integration in Anlagen zur Verfügung stellen müssen. Der gesamte Cost of Ownership, sprich die realen Kosten über den Lifecycle einer Anlage hinweg, ist zu reduzieren. Eine Herabsetzung der

laufenden Betriebskosten zum Nachteil der Investitionskosten bringt für den Betreiber nur eine Verschiebung und keine ökonomische Verbesserung.

Eine detaillierte Analyse der vorhandenen Engineering-Konzepte und Technologien konnte zur Lösungsfindung beitragen. Forderungen aus dem Bereich der Verfahrenstechnik nach einer Vereinfachung der Handhabung von Anlagenteilen wurden im Konzept berücksichtigt. Um sowohl den Schulungsaufwand seitens der Betreiber zu reduzieren als auch die Akzeptanz einer derartigen Lösung zu maximieren, erfolgte die Einbeziehung geeigneter Konzepte und Technologien, wie das DeKOS-Bedienmodell und FDT als Proxy-Technologie, bei der Umsetzung des Lösungsansatzes.

Nach der Untersuchung des Nutzens einer standardisierten Hilfsfunktionalität auf Systemebene für den Anlagenbetreiber wurde der Bedarf nach einem durchgängigen Engineering-Konzept basierend auf einem einheitlichen Modell belegt. Durch die Bewertung, Auswahl und Erweiterung der bereits existierenden Referenzarchitektur DRA[86] wurde die Basis eines offenen, plattformunabhängigen Modellierungskonzeptes zur Kapselung der Hilfsfunktionalität von Partialsystemen festgelegt. Durch den Ansatz der Strukturierung der standardisierten Hilfsfunktionen wurde der Forderung nach einem vereinfachten durchgängigen Engineering-Konzept zur Realisierung der Bedienobjekte entsprochen. Beide Partialmodelle sind unabhängig von Hersteller, Geräte- und Feldbustyp sowie der Partialsystemkonstellation. Die Spezifikation der erforderlichen Modelle liegt in Form von UML-Klassendiagrammen vor.

Auf Basis der spezifizierten Partialmodelle StrucRefArchitecture und SysRefArchitecture erfolgt die formale Notation. Hierzu wird die Metasprache XML verwendet. Durch die Erstellung der XML-Schema-Definitionen sind die Modellierungskonzepte formal beschrieben und bilden die Vorlage zur Erstellung und Validierung von Partialmodellen. Diese können als Instanzen in entsprechenden Softwaretools verarbeitet und zur komfortablen Erzeugung des Proxy-Quellcodes genutzt werden. Dabei bietet das XML-Umfeld einen weiteren potenziellen Mehrwert, da diverse Tools, wie beispielsweise Stylesheet Transformation Processors, Parser etc. zur freien Verfügung stehen und sich somit die Realisierung entsprechender Software-Komponenten einfach gestaltet.

Die Arbeit beinhaltet abschießend die Anwendung des Modellierungskonzeptes. Die prototypisch realisierten Softwaretools zur komfortablen Erstellung konformer Systembeschreibungen werden erläutert. Hierdurch erfolgt eine Unterstützung des gesamten Engineering-Prozesses von der Erstellung passender Gerätebeschreibungen bis hin zur Generierung des Proxy-Quellcodes. Nach einer Einführung in die Grundlagen der Software-Architektur der implementierten

[86] DeKOS Referenz Architektur

Proxy-Lösung auf Basis der FDT-Technologie endet die Arbeit mit der Realisierung eines Bedienobjektes für eine Pumpstation als Beispielanwendung.

7.2 Bewertung der Lösung

Primäres Ziel der Arbeit ist, das Anlagenpersonal bei der Handhabung komplexer Automatisierungssysteme zu unterstützen, sekundär ist eine Hilfestellung für den Anlagenbauer und Gerätehersteller zu leisten. Wie zuvor beschrieben soll durch die Lösung eine Kostenreduzierung und keine Kostenverschiebung stattfinden. Ein wichtiges Kriterium bei der Erstellung des Lösungskonzeptes ist dabei die Prämisse, die Vorteile bestehender offener Technologien unter keinen Umständen einzuschränken. Zur Beurteilung der Lösung erfolgt eine detaillierte Validierung gegenüber den Anforderungen aus Kapitel 2.

Das erzielte Ergebnis erleichtert dem Hersteller die Spezifikation der Hilfsfunktionalität neuer Geräte, indem ihm sowohl eine Funktionsbibliothek mit einheitlichem Namensraum und präziser semantischer Definition zur Auswahl gestellt wird, als auch diese in einer übersichtlichen strukturierten Form angeboten werden. Dabei ist durch das einheitliche Bedienmodell sichergestellt, dass unabhängig von dem Hersteller der eingesetzten Geräte eine Weiterverarbeitung der Gerätebeschreibung für Partialsystemebene gegeben ist. Durch die vorgegebene Implementierung des Bedienobjektes als DTM mit erweiterten Objekteigenschaften ist gewährleistet, dass die Gerätefunktionalität ohne zusätzlichen Implementierungsaufwand vom Bedienobjekt eines Partialsystems genutzt werden kann. Durch die Verwendung einer etablierten Technologie ergibt sich für den Gerätehersteller kein zusätzlicher Aufwand, da er ohnedies für den Verkauf seiner Geräte geeignete Bedienobjekte anbieten muss. Die Software-Architektur sieht wieder verwendbare Teile bei der Implementierung eines DTMs vor. Durch die flexiblen Erweiterungsmöglichkeiten der Strukturierungssichten ist der Hersteller in der Lage, auch die angebotenen Hilfsfunktionen der Benutzerschnittstelle nach seinen eigenen Bedürfnissen strukturiert anzubieten. Dabei ist der Anwender nicht an eventuelle Vorgaben bei der Bedienung gebunden, da ein beliebiges Wechseln zwischen den einzelnen Sichtweisen jederzeit durchführbar ist.

Der Anlagenbauer ist in der Lage, aufbauend auf den konformen Gerätebeschreibungen, Bedienobjekte für Partialsysteme zu generieren. Der hierzu erforderliche Engineering-Aufwand ist minimal. Durch die Definition der Systemgrundfunktionen kann auf Partialsystemebene eine große Menge an Hilfsfunktionen dem Anwender automatisch zur Verfügung gestellt werden. Durch die festgelegte Abarbeitungsvorschrift im Modell ist weder ein zusätzlicher Implementierungsaufwand noch eine Entscheidung über die benötigte Funktionalität

notwendig. Der Implementierungsaufwand beschränkt sich weitestgehend auf die Wiederverwendung von Softwarekomponenten. Die Benutzerschnittstelle wird äquivalent zur Geräteebene aus den Informationen der DDD[87] des Partialsystems beim Start des Bedienobjektes generiert.

Dem Anlagenpersonal ermöglicht die vorgestellte Lösung eine einheitliche Sichtweise und einheitliche Bedienung von einzelnen Geräten und Anlagenteilen im Bereich der Fertigungs- und Verfahrenstechnik. Dabei ist auf jeder Hierarchieebene der Anlage die Bedienphilosophie einheitlich. Darüber hinaus existiert ausschließlich ein einheitlicher Namensraum mit semantisch präzisen Hilfsfunktionen. Das erforderliche Wissen, eventuelle Abhängigkeiten, Parametersätze etc., die bei der Handhabung benötigt werden, bleiben durch die Kapselung im Bedienobjekt des Partialsystems verborgen. Durch die modellbasierte Generierung des Bedienobjektes ist es dem Betreiber auch ohne Spezialwissen nachträglich möglich, Hilfsfunktionen auf Partialsystemebene, wie beispielsweise Asset Management Funktionen zur Diagnose, ergänzend hinzuzufügen. Hierzu wird assistentengestützt die DDD des Systems erweitert und anschließend der Kern des Bedienobjektes neu generiert. Nach einem Neustart des DTMs steht anschließend die zusätzliche Hilfsfunktion dem Personal zur Verfügung.

Zusammenfassend kann festgestellt werden, dass der Einsatz der in dieser Arbeit spezifizierten Lösungskomponenten und Vorgehensweisen sowohl zu einem Mehrwert seitens des Personals bei der Anlagenbedienung führt, als auch den hierzu erforderlichen Engineering-Aufwand erheblich reduziert. Die vorliegende formale Spezifikation stellt dabei die eindeutige Darstellung der Hilfsfunktionalität und der Abhängigkeiten sicher. Eine herstellerneutrale Verwendung ist hierdurch gegeben.

7.3 Ausblick

Der Schwerpunkt des vorgestellten Lösungsweges besteht darin, den Aufwand für das Erstellen entsprechender Bedienobjekte auf Partialsystemebene zu reduzieren, denn die anwenderfreundlichsten Lösungen sind zwar oft proprietär mit hohem Aufwand realisierbar, stehen aber in keinem Verhältnis zum jeweiligen Mehrwert. Aufbauend auf dem spezifizierten Modell wäre eine vollständig automatische Generierung des Bedienobjektes eines Partialsystems denkbar. Das entwickelte Modell beinhaltet die Systemgrundfunktionen mit einer vorgeschriebenen Abarbeitungslogik. Neben dieser Menge der fraktalen Funktionen existieren, wie in der Arbeit beschrieben, eine Vielzahl an polymorphen Funktionen. Bei diesen hängt die Art und Weise der Abarbeitung der Funktion auf Par-

[87] DeKOS Device Description

tialsystemebene von der Konstellation der unterlagerten Geräte ab. Es wäre eventuell denkbar, für spezielle Kombinationen von Geräten die Abarbeitungslogik zu untersuchen. So könnte eine Bibliothek von standardisierten Kombinationen von Feldgeräten realisiert werden, bei denen die Systemlogik durch die Gerätekonstellation festgelegt ist.

Das in Kapitel 6.1 vorgestellte Softwaretool zur Erstellung der Beschreibungen könnte dahingehend erweitert werden, dass in Anlehnung an SFC- und CFC-Pläne der [IEC 61131-3] eine graphische Verschaltung der Hilfsfunktionen vom Partialsystem und der unterlagerten Feldgeräte möglich ist. Hierdurch wäre eine weitere Reduzierung des Aufwands bei der Erstellung der Systembeschreibung zu erzielen. Aus den graphischen Verschaltungsinformationen könnten so die notwendigen Algorithmen für die Abarbeitung der unterlagerten Hilfsfunktionen gewonnen werden.

Zusätzlich kann der Implementierungsaufwand durch den Einsatz von Geräteprofilen gesenkt werden. Durch das jeweilige Profil ist eine eindeutige Zuordnung der Parameter zur standardisieren Hilfsfunktion möglich, somit ist eine vollständige Generierung eines DeKOS-DTMs auf Basis der Geräte-DDD denkbar.

Essentieller Bestandteil der vorgestellten Lösung ist nach heutigem Stand der Technik der Einsatz von Proxy-Technologien. Sofern Feldgeräte der nächsten Generation über ausreichend Rechenkapazität verfügen und der Paradigmenwechsel zu einer Implementierung der standardisierten Hilfsfunktionalität in den jeweiligen Feldgeräten führt, kann auf die Umsetzung der parameterorientierten Geräte auf die funktionsorientierte Sichtweise durch eine Proxy-Lösung verzichtet werden. Des Weiteren ist das spezifizierte Modell technologieunabhängig realisiert. Die Wahl fiel zugunsten der etablierten FDT-Technologie aus. Hierdurch ist eine Integration der Lösung in jedes moderne Prozessleitsystem gewährleistet. Darüber hinaus existieren diverse Stand-alone-Anwendungen mit FDT-Schnittstelle, die im Bereich der Fertigungstechnik oder auch bei Kleinanlagen der Verfahrenstechnik eingesetzt werden können, um die beschriebene Lösung umzusetzen.

Literaturverzeichnis

Ahrens/Scheurlen/Spohr 97	Ahrens, W.; Scheurlen, H.-J.; Spohr, G.-U.: *Informationsorientierte Leittechnik*. Oldenbourg-Verlag, München, Wien: 1997.
Anderl 02	Anderl, R.: *Produktdatentechnologie 1*. Hypermediales Skriptum zur Vorlesung. Internetquelle URL: http://www.iim.maschinenbau.tu-darmstadt.de /pdt1/frames/PDT1.html.
Bacher 96	Bacher, J.: *Clusteranalyse - Anwendungsorientierte Einführung*. 2. Auflage, Oldenbourg-Verlag, München, Wien: 1996.
Backhaus et al. 94	Backhaus, K. et al.: *Multivariate Analysemethoden - Eine anwendungsorientierte Einführung.* 7. Auflage, Springer-Verlag, Berlin, Heidelberg: 1994.
Balzert 95	Balzert, H.: *Methoden der objektorientierten Systemanalyse*. B.I.-Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich: 1995.
Becker 01	Becker, N.: *Das Prozessautomatisierungssystem DelatV von Fisher-Rosemount*. Automatisierungstechnische Praxis atp 1/2001, S. 28-40.
Bender 02	Bender, K.: *Automatisierungstechnik im Maschinenwesen.* Vorlesungsskriptum 02/03, Technische Universität München: 2002.
Bender et al. 02	Bender, K. et al.: *DEKOS-Engineering-Modell für intelligente Automatisierungskomponenten*. Utz-Verlag, München: 2002.
Bender/Kuttig/Meyer 01	Bender, K.; Kuttig, F.; Meyer, H.: *Continuous Engineering Through Uniform Device-Functions and Standard Web-Technologies*. International Conference on Fieldbus Systems and their Applications, Nancy 2001, S. 83-88.

Bender/Kuttig/Meyer 02	Bender, K.; Kuttig, F.; Meyer, H.: *Intelligentes Engineering mittels einheitlicher Technologien*. Automatisierungstechnische Praxis atp 1/2002, S. 32-38.
Bender/Meyer 02	Bender, K.; Meyer, H.: *4 Jahre DeKOS - ein kurzes Fazit*. MessTec&Automation, 10/2002, S. 72.
Birkhofer 01	Birkhofer, R.: *Modellbasierte Beschreibung zur offenen Integration intelligenter Feldgeräte der Automatisierungstechnik*. Utz-Verlag, München: 2001.
Blum 02	Blum, M.: *Standardisierte Beschreibung intelligenter mechatronischer Systeme*. Diplomarbeit, Technische Universität München: 2002.
Booch 99	Booch, G. et al.: *Das UML-Benutzerhandbuch*. Addison Wesley, Bonn, Menlo Park: 1999.
Bregulla 01	Bregulla, M.: *Bedienobjekte mit standardisierter Funktionalität für busgekoppelte Automatisierungskomponenten*. Utz-Verlag, München: 2001
Brehm 03	Brehm, A.: *Praktikum der technischen Chemie - Wärmeübertragung*. Universität Oldenburg: 2003, S. 1-15.
Breutmann/Burkhardt 92	Breutmann, B.; Burkhardt, R.: *Objektorientierte Systeme*. Hanser-Verlag, München, Wien: 1992.
DIN 19222	DIN 19222: *Leittechnik - Begriffe*. Beuth-Verlag, Berlin: 2001.
DIN 19227	DIN 19227: *Leittechnik - Graphische Symbole und Kennbuchstaben für die Prozeßleittechnik, Darstellung von Aufgaben*. Teil 1, Beuth-Verlag, Berlin: 1993.
DIN 19233	DIN 19233: *Prozessautomatisierung - Begriffe.* Beuth-Verlag, Berlin: 1998.
DIN 28004	DIN 28004: *Chemischer Apparatebau - Dokumentenart im Lebensweg von Prozessanlagen*. Teil 1, Beuth-Verlag, Berlin: 2002.
DIN 31051	DIN 31051: *Grundlagen der Instandhaltung*. Draft-Version, Beuth-Verlag, Berlin: 2001.

DIN 66201	DIN 66201: *Technischer Prozess - Definitionen*. Beuth-Verlag, Berlin: 1997.
Dressler/Trilling 96	Dressler, T.; Trilling, U.: *Instandhaltung der prozessnahen Technik - Cost of Ownership*. Automatisierungstechnische Praxis atp 2/1996, S. 42-46.
Duden 01	Duden: *Informatik.* Dudenverlag, Mannheim, Wien, Zürich: 2001.
Eddon/Eddon 98	Eddon, G.; Eddon, H.: *Inside Distributed COM*. Microsoft Presse Deutschland, Unterschleißheim: 1998.
E+H 03	Endress+Hauser: *Fieldcare Plant Asset Management*. Internetquelle URL: http://www.endress.com.
EN 13306	DIN EN 13306: *Begriffe der Instandhaltung*. Beuth-Verlag, Berlin: 2001.
EN 50170	DIN EN 50170: *Die europäische Feldbusnorm*. Beuth-Verlag, Berlin: 1996.
Enste/Hübner 03	Enste, U.; Hübner,I.: *Perspektiven der Leittechnik*. etz 1-2/2003, S. 33-34.
Epple 03	Epple, U.: *Online Asset Management*. Internetquelle URL: http://www.plt.rwth-aachen.de/am/am.html.
Eversheim 90	Eversheim, W.: *Inbetriebnahme komplexer Maschinen und Anlagen*. VDI-Verlag, Düsseldorf: 1990.
Eversheim/Schuh 96a	Eversheim, W.; Schuh, G.: *Betriebshütte - Produktion und Management – Teil 1*. 7. Auflage, Springer-Verlag, Berlin, Heidelberg, New York: 1996.
Eversheim/Schuh 96b	Eversheim, W.; Schuh, G.: *Betriebshütte - Produktion und Management – Teil 2*. 7. Auflage, Springer-Verlag, Berlin, Heidelberg, New York: 1996.
Früh 97	Früh, K.-F.: *Handbuch der Prozessautomatisierung*. Oldenbourg-Verlag, München, Wien: 1997.

Grossmann 02	Grossmann, D.: *Entwicklung einer universellen Kommunikationsschnittstelle für Bedienproxys*. Semesterarbeit, Technische Universität München: 2002.
Grossmann 03	Grossmann, D.: *Konzept einer applikationsorientierten Anlagenstrukturierung zur webbasierten Systemdiagnose*. Diplomarbeit, Technische Universität München: 2003.
Henning/Kutscha 94	Henning, K.; Kutscha, S.: *Informatik im Maschinenbau*. 4. Auflage, Springer-Verlag, Berlin, Heidelberg, New York: 1994.
Hoang/Rieger 99	Hoang, M.-S.; Rieger, P.: *Komponentenbasierte Automatisierungssoftware*. Hanser-Verlag, München, Wien: 1999.
IEC 60050-191	IEC 60050-191: *International Electrotechnical Vocabulary - Chapter 191: Dependability and quality of service*.
IEC 61131-3	IEC 61131-3: *Speicherprogrammierbare Steuerungen - Teil 3, Programmiersprachen*.
ISO 8879	ISO 8879: *Information processing - Text and office systems - Standard Generalized Markup Language (SGML)*. Internation Organization for Standardization: 1986.
itm 02a	itm: *Tagungsband DeKOS Abschlussveranstaltung*. Technische Universität München: 2002.
itm 02b	itm: *DeKOS Spezifikation V1.2*. Technische Universität München: 2002.
Kiencke 97	Kiencke, U.: *Ereignisdiskrete Systeme - Modellierung und Steuerung verteilter Systeme*. Oldenbourg-Verlag, München, Wien: 1997.
Kieß 95	Kieß, J.: *Objektorientierte Modellierung von Automatisierungssystemen - Software Engineering für Embedded Systems*. Springer-Verlag, Berlin, Heidelberg, 1995.

Koschel/Müller/Nicklaus 00	Koschel, J.; Müller, U.; Nicklaus, E.: *Asset Management - Zustandserkennung in Produktionsanlagen*. Automatisierungstechnische Praxis atp 12/2000, S. 28-40.
Kuchling 88	Kuchling, H.: *Taschenbuch der Physik*. Harri Deutsch Verlag, 10. Auflage, Thun, Frankfurt a. M.: 1988.
Kuttig/Meyer 02	Kuttig, F.; Meyer, H.: *Einheitliches Geräte-Interface durch FDT - Praktikabel auch für die Industrieautomation*. IEE - Automatisierung und Datentechnik 3/2002, S. 58-60.
Kuttig/Meyer/Zirkler 02	Kuttig, F.; Meyer, H.; Zirkler, A.: *Intelligenz im Feld - Realisierung des Engineering-Zugriffs mit FDT* . MessTec&Automation, 11/2002, S. 66-67.
Lauber/Göhner 99a	Lauber, R.; Göhner, G.: *Prozessautomatisierung 1*. Springer-Verlag, Berlin, Heidelberg, New York: 1999.
Lauber/Göhner 99b	Lauber, R.; Göhner, G.: *Prozessautomatisierung 2*. Springer-Verlag, Berlin, Heidelberg, New York: 1999.
Lindner 03	Lindner, K.-P.: *Asset Management - Ein Allheilmittel für den Kostendruck*. GMA-Kongress 2003, VDI-Berichte 1756, VDI-Verlag, Düsseldorf: 2003, S. 893-900.
Lobin 00	Lobin, H.: *Informationsmodellierung in XML und SGML*. Springer-Verlag, Berlin, Heidelberg, New York: 2001.
Mandelbrot 87	Mandelbrot, B.: *Die fraktale Geometrie der Natur*. Birkhäuser-Verlag, Biel-Benken: 1987.
Meier 02	Meier, P.: *Vertikale Integration in der Automatisierungstechnik unter Verwendung objektorientierter Middleware*. Semesterarbeit, Technische Universität München: 2002.
Meyer 01	Meyer, H.: *Intelligentes Geräte-Engineering mit PACTware und FDT*. MessTec 12/2001, S. 53-54.

Meyer 03	Meyer, H.: *Diagnose von mechatronischen Systemen durch einheitliche Bedienobjekte*. GMA-Kongress 2003, VDI-Berichte 1756, VDI-Verlag, Düsseldorf: 2003, S. 263-270.
Meyer/Bender 01	Meyer, H.; Bender, K.: *Subsystemfunktionalität auf Basis spezifizierter Bedienfunktionen für Feldgeräte*. SPS/IPC/DRIVES 2001, Hüthig-Verlag, Heidelberg: 2001, S. 99-107.
Meyer/Bregulla/Bender 01a	Meyer, H.; Bregulla, M.; Bender, K.: *Einheitliche Gerätefunktionen für ein durchgängiges Engineering am Beispiel einer FDT/DTM-Implementierung*. GMA-Kongress 2001, VDI-Berichte 1608, VDI-Verlag, Düsseldorf: 2001, S. 545-552.
Meyer/Bregulla/Bender 01b	Meyer, H.; Bregulla, M.; Bender, K.: *Einheitliche Gerätefunktionen für intelligente Feldbusgeräte*. MM Maschinenmarkt 13/2001, S. 52-55.
Müller et al. 02	Müller, J. et al.: *Asset Management Box*. Tagungsband AKIDA 2002, Aachener Beiträge zur angewandten Rechnertechnik, Band 46, Mainz-Verlag, Aachen 2002.
Näf 02	Näf, M.: *Einführung in XML, DTD und XSL*. Internetquelle URL: http://www.internet-kompentenz.ch/xml/.
NE 57	NE 57: *Anzeige- und Bedienoberfläche für statische Frequenzumrichter*. NAMUR-Gesellschaft, Leverkusen: 1995.
NE 91	NE 91: *Anforderungen an Systeme für anlagennahes Asset Management*. NAMUR-Gesellschaft, Leverkusen: 2001.
Nicklaus/Fuß 00	Nicklaus, E.; Fuß, H.-P.: *Online Asset Management*. Automatisierungstechnische Praxis atp 5/2000, S. 30-39.

Nothnagel 99	Nothnagel, M.: *Klassifikationsverfahren der Diskriminanzanalyse - Eine vergleichende und integrierende Übersicht*. Diplomarbeit, überarbeitete Fassung, Humboldt Universität Berlin: 1999.
Nunge 01	Nunge, S.: *Der Referenzanlagenansatz zur Ableitung von Luftreinhaltestrategien.* Dissertation, VDI-Verlag, Universität Karlsruhe: 2001.
PNO 00	PROFIBUS Guideline: *DTM Stylguide - Guideline for the implementation of Device Type Managers (DTMs) for field devices V1.1*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 2000.
PNO 01a	PROFIBUS Guideline: *PROFInet Implementation Guide V1.0*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 2001.
PNO 01b	PROFIBUS Guideline: *Specification for PROFIBUS Device Description and Device Integration - Volume 3, FDT V1.2*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 2001.
PNO 02a	PROFIBUS: PROFInet *Technologie und Anwendung - Systembeschreibung*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 2002.
PNO 02b	PROFIBUS: *EDDL - Einheitliches Gerätemanagement*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 2002.
PNO 02c	PROFIBUS International: *PROFInet - Einsatz im Feld*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 2002.
PNO 99	PROFIBUS Profile: *PROFIBUS-PA Profile for Process Control Devices*. PROFIBUS Nutzerorganisation e. V., Karlsruhe: 1999.
Polke 92	Polke, M.: *Prozessleittechnik.* Oldenbourg-Verlag, München, Wien: 1992.
Richter 03	Richter, H.: *Internationales Concurrent Engineering: Anforderungen an die Engineering-Systeme aus Sicht des Anlagenbauers*. Automatisierungstechnische Praxis atp 3/2003, S. 36-39.

Rieger 95	Rieger, B.: Systematische Funktionsüberprüfung bei der Inbetriebnahme feldbusvernetzter Automatisierungssysteme. VDI-Verlag, Düsseldorf: 1995.
Römer 02	Römer, M.: *Grundlagenseminar Profibus*. Technische Universität München: 2002.
Scherf/Haese/Wenzek 99	Scherf, B.; Haese, E.; Wenzek, H.: *Feldbussysteme in der Praxis - Ein Leitfaden für den Anwender*. Springer-Verlag, Berlin, Heidelberg, New York: 1999.
Schnell 99	Schnell, G.: *Bussysteme in der Automatisierungstechnik - Grundlagen und Systeme der industriellen Kommunikation*. 3. Auflage, Vieweg-Verlag, Braunschweig, Wiesbaden: 1999.
Schnieder 01	Schnieder, E.: *Engineering komplexer Automatisierungssysteme*. EKA 2001, Technische Universität Braunschweig: 2001.
Schnieder 99	Schnieder, E.: *Methoden der Automatisierung*. Vieweg-Verlag, Braunschweig, Wiesbaden, 1999.
Schöning 00	Schöning, U.: *Logik für Informatiker*. Spektrum, 5. Auflage, Heidelberg 2000.
Schraft/Kaun 98	Schraft, R.-D.; Kaun, R.: *Automatisierung in der Produktion - Erfolgsfaktoren und Vorgehen in der Praxis*. Springer-Verlag, Berlin, Heidelberg, New York: 1998.
Schuler 99	Schuler, H.: *Prozessführung.* Oldenbourg-Verlag, München, Wien: 1999.
Schullerer 01	Schullerer, J.: *PACTware - im Umfeld FDT und Anlagenleitsystem*. Automatisierungstechnische Praxis atp S/2001, S. 21-26.
Schultz-Wild/Lutz 97	Schultz-Wild, L.; Lutz, B.: *Industrie vor dem Quantensprung*. Springer-Verlag, London, Berlin, Heidelberg: 1997.
Simon 03	Simon, R.: *Einheitliches Konfigurieren*. Computer & Automation 3/2003, S. 36-41.
Sperber 03	Sperber, M.: *Basis des PC-based Control*. Computer&Automation 07/2003, S. 16-18.

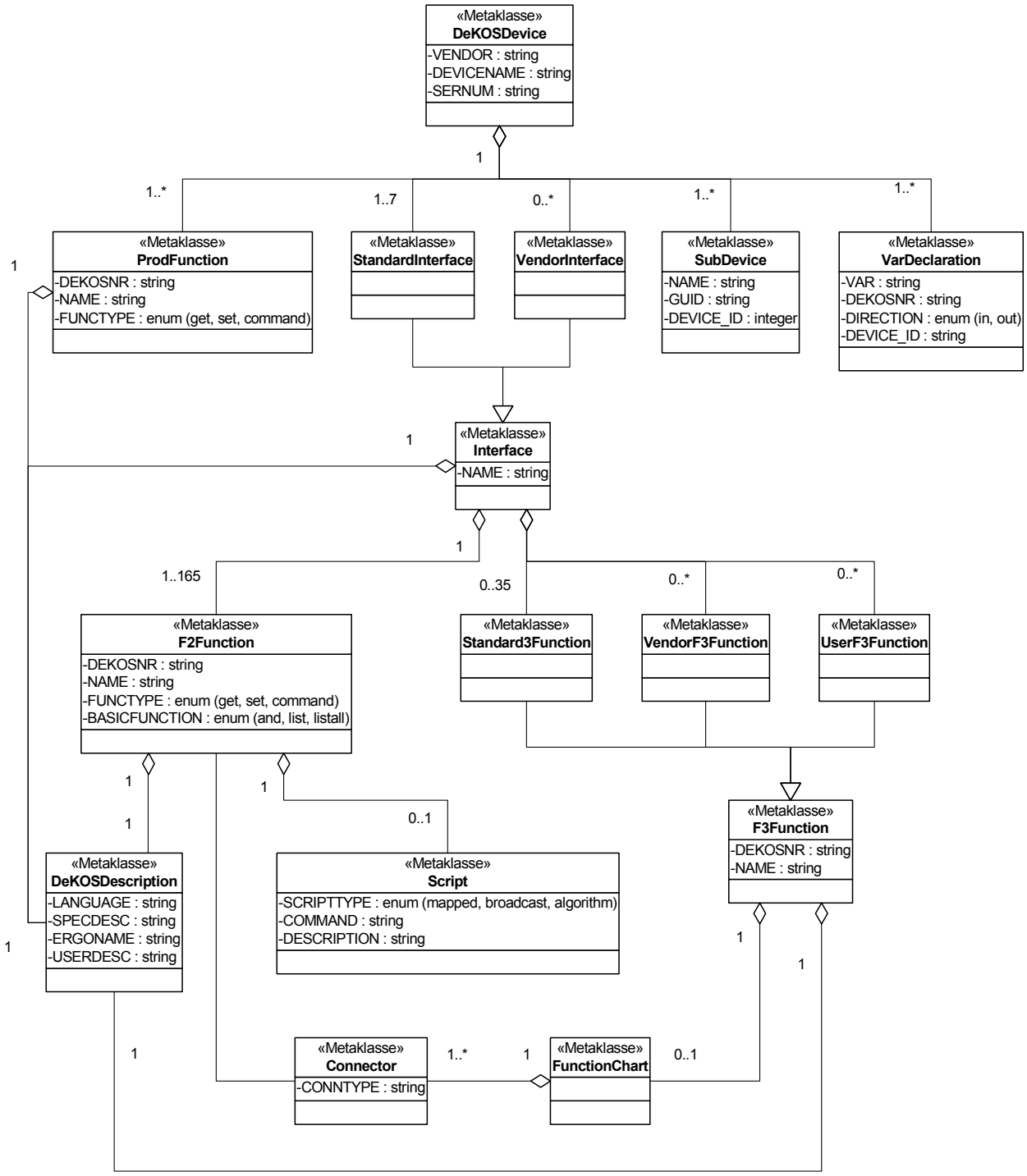
Strohrmann 98	Strohrmann, G.: *Automatisierungstechnik 1 - Grundlagen, analoge und digitale Prozessleittechnik*. 4. Auflage, Oldenbourg-Verlag, München, Wien: 1998.
Stupp/Oestreich 03	Stupp, F.; Oestreich, V.: *Der Einzug von PROFIBUS*. Computer & Automation 3/2003, S. 42-46.
Tomasik 01	Tomasik, P.: *Einsatzmöglichkeiten standardisierter DeKOS-Bedienfunktionalität in der täglichen Praxis*. DeKOS Vollversammlung, Technische Universität München: 2001.
Tomaszunas 98	Tomaszunas, J.: *Komponentenbasierte Maschinenmodellierung zur Echtzeit-Simulation für den Steuerungstest*. Utz-Verlag, München:1998.
VDI 2186	VDI/VDE 2186: *Einheitliche Anzeige- und Bedienoberfläche für Antriebsregelgeräte*. Beuth-Verlag, Berlin: 2000.
VDI 2187	VDI/VDE 2187: *Einheitliche Anzeige- und Bedienoberfläche für digitale Feldgeräte*. Beuth-Verlag, Berlin: 2002.
VDI 3633	VDI/VDE 3633: *Simulation von Logistik-, Materialfluss- und Produktionssystemen - Begriffsdefinitionen*. Beuth-Verlag, Berlin: 1996
W3C 03	World Wide Web Consortium: *Extensible Markup Language (XML) V1.0 (Second Edition)*. Internetquelle URL: http://www.w3c.org/TR/2000/REC-xml-20001006.
Walther 90	Walther, E.: *Taschenbuch technischer Formeln*. Harri Deutsch-Verlag, Thun, Frankfurt am M.: 1990.
Warnecke 95	Warnecke, H.-J.: *Aufbruch zum fraktalen Unternehmen - Praxisbeispiele für neues Denken und Handeln.* Springer-Verlag, Berlin, Heidelberg: 1995.
Weber 96	Weber, K.: *Inbetriebnahme verfahrenstechnischer Anlagen - Vorbereitung und Durchführung*. Springer-Verlag, Berlin, Heidelberg, New York: 1996.

Abbildungsverzeichnis

Tabellenverzeichnis

Stichwortverzeichnis

Anhang A Klassenmodell SysRefArchitecture



Anhang B Gerätegrundfunktionen

Atomare Hilfsfunktion	*Beschreibung*
`backupDeviceData`	Gerätedaten, -Parameter, Speicherblöcke etc. werden auf Datenträger gesichert.
`checkDevice`	Leitet eine Überprüfung der Feldgeräte-Hardware ein, führt also eine Art Selbstdiagnose bezüglich Prozessor, Speicher etc. aus.
`getAlarmInfo`	Liest Informationen zu einem Alarm aus dem Feldgerät aus (Textuelle Informationen, Status_quittiert/unquittiert, anstehend/nicht anstehend, aktiv/inaktiv). Dieser wird vom Prozess ausgelöst (Prozessgröße).
`getAlarmMask`	Liest aus, welche Alarme das Gerät unterstützt und welche aktiviert sind.
`getAlarmParameters`	Liest Grenzwerte/Bedingungen für die Alarmauslösung ein.
`getCommunicationParameters`	Liest Busparameter des Feldgerätes aus.
`getCountry`	Liest die aktuelle Landeseinstellung aus.
`getDeviceError`	Liest die Fehlermeldungen aus dem Feldgerät aus.
`getDeviceErrorMask`	Liest die Fehlermeldungen die das Feldgerät unterstützt aus.
`getDeviceErrorParameters`	Liest Parameter, die bei der Auslösung und Behandlung von Fehlern in dem Feldgerät eingestellt sind; z.B. max. Strom für Binärausgang, bei dem eine Fehlermeldung ausgelöst werden soll etc.
`getDeviceWarning`	Liest eine Warnmeldung aus dem Feldgerät aus, (Textuelle Informationen, Status_ quittiert/unquittiert, anstehend/nicht anstehend, aktiv/inaktiv).
`getDeviceWarningMask`	Liest die unterstützten Warnungen des Feldgerätes aus und zeigt an, welche aktiviert sind.
`getDeviceWarningParameters`	Liest Parameter, die bei der Auslösung und Behandlung von Warnungen, die das Feldgerät betreffen, eingestellt sind; z.B. Anzahl von Speicherfehlern nach denen eine Warnung ausgelöst werden soll, etc.

`getFailSafeParameters`	Liest das eingestellte Verhalten beim Übergang in den Failsafe-Zustand aus z. B. Sollwert, Sicherheitsposition etc. (PNO_FSAFE_TYPE).
`getIdentification`	Liest Identifikationsdaten aus dem Feldgerät aus.
`getLanguage`	Liest die Spracheinstellung für das Feldgerät aus.
`getMaintenanceData`	Liest Instandhaltungsdaten aus dem Feldgerät aus.
`getOperatingTime`	Liest Daten aus dem Feldgerät, die die Betriebsdauer betreffen.
`gotoFailSafe`	Schaltet die Ausgänge des Feldgerätes in den sicheren Zustand.
`login`	Am Feldgerät/Proxy anmelden.
`logout`	Am Feldgerät/Proxy abmelden.
`resetAlarm`	Quittiert Alarmmeldungen.
`resetAllExceptions`	Quittiert alle Alarme, Warnungen, Hard- und Softwarefehler und -warnungen.
`resetDevice`	Warmstart des Feldgerätes einleiten.
`resetDeviceError`	Quittiert Gerätewarnungen.
`resetDeviceWarning`	Quittiert Gerätefehler.
`resetOperatingTime`	Setzt den Betriebsstundenzähler zurück.
`resetToDefault`	Rücksetzen des Gerätes auf die Werkseinstellung. Falls einzelne Komponenten zurückgesetzt werden können, ist die Angabe dieser möglich.
`restoreDeviceData`	Gerätedaten, -Parameter, Speicherblöcke etc. werden in das Gerät zurück geschrieben.
`setAlarmKey`	Setzt eine Gerätekennung, die einer Alarmmeldung des Feldgerätes beigefügt wird.
`setAlarmMask`	Setzen und Aktivieren der Alarme die das Gerät unterstützt.
`setAlarmParameters`	Stellt Grenzwerte/Bedingungen für Alarmauslösung ein.
`setCommunicationParameters`	Stellt Feldbusparameter des Feldgerätes ein.
`setCountry`	Stellt das Land ein, dessen Maßeinheiten etc. gelten sollen.
`setDeviceErrorMask`	Setzt die Fehlermeldungen die das Feldgerät

	unterstützt.
`setDeviceErrorParameters`	Schreibt Parameter, die zur Auslösung und Behandlung von Fehlermeldungen im Feldgerät notwendig sind, z.B. max. Strom für Binärausgang bei dem eine Fehlermeldung ausgelöst werden soll etc.
`setDeviceWarningMask`	Aktiviert/deaktiviert die unterstützten Warnungen des Feldgerät.
`setDeviceWarningParameters`	Schreibt Parameter, die zur Auslösung und Behandlung von Warnungen, die das Feldgerät betreffen, notwendig sind; z.B. Anzahl von Speicherfehlern nach denen eine Warnung ausgelöst werden soll etc.
`setFailSafeParameters`	Setzt das Verhalten beim Übergang in den Failsafe-Zustand z. B. Sollwert, Sicherheitsposition etc.
`setIdentification`	Setzt Identifikationsdaten des Feldgerätes.
`setLanguage`	Stellt die Spracheinstellungen für das Feldgerät ein
`setMaintenanceData`	Schreibt die Instandhaltungsdaten in das Feldgerät.

Anhang C Standardstrukturierungssicht

Bedienung	
Anweisung	
gotoFailSafe	Atomarfunktion
resetAlarm	Atomarfunktion
resetAllExceptions	Atomarfunktion
resetApplication	Atomarfunktion
resetApplicationError	Atomarfunktion
resetApplicationWarning	Atomarfunktion
resetDevice	Atomarfunktion
resetDeviceError	Atomarfunktion
resetDeviceWarning	Atomarfunktion
selectInitMode	Atomarfunktion
selectParametersSet	Atomarfunktion
setOperatingMode	Atomarfunktion
startApplication	Atomarfunktion
stopApplication	Atomarfunktion
Anzeigen	
browseApplication	Atomarfunktion
browseApplicationData	Atomarfunktion
browseDeviceIdentification	Komfortfunktion
browseDeviceInformation	Komfortfunktion
browseDevicespecificFunctions	Atomarfunktion
browseMotionSets	Komfortfunktion
browseOperatingState	Komfortfunktion
browseOperatingvalues	Komfortfunktion
browseProcessInformation	Komfortfunktion
browseValueInformation	Komfortfunktion
getAuxValue	Atomarfunktion
getCertification	Atomarfunktion
getComplianceInfo	Atomarfunktion
getControlType	Atomarfunktion
getFixedValues	Atomarfunktion
getIdentification	Atomarfunktion
getIOValue	Atomarfunktion
getOperatingMode	Atomarfunktion
getOperatingTime	Atomarfunktion
Zugangsberechtigung	
login	Atomarfunktion
logout	Atomarfunktion
Projektierung	
Integration	
assingCommunication	Komfortfunktion

browseCommunication	Komfortfunktion
getCommunicationParameters	Atomarfunktion
getCountry	Atomarfunktion
getLanguage	Atomarfunktion
setCommunicationParameters	Atomarfunktion
setCountry	Atomarfunktion
setIdentification	Atomarfunktion
setLanguage	Atomarfunktion
Konfigurierung	
Einheiten/Auflösung	
getDecPointPos	Atomarfunktion
getMeasurementDimension	Atomarfunktion
getMeasurementResolution	Atomarfunktion
getUnit	Atomarfunktion
setDecPointPos	Atomarfunktion
setMeasurementDimension	Atomarfunktion
setMeasurementResolution	Atomarfunktion
setUnit	Atomarfunktion
Messwertaufnahme	
getMaxMeasurementValue	Atomarfunktion
getMeasurementOffsetValue	Atomarfunktion
getMinMeasurementValue	Atomarfunktion
getSensorLimits	Atomarfunktion
resetMeasurementOffsetValue	Atomarfunktion
setMaxMeasurementValue	Atomarfunktion
setMeasurementOffsetValue	Atomarfunktion
setMinMeasurementValue	Atomarfunktion
setSensorLimits	Atomarfunktion
Schnittstelle	
assignIOConfiguration	Komfortfunktion
browseIOConfiguration	Komfortfunktion
getDataExchangeMask	Atomarfunktion
getMaxOuputValue	Atomarfunktion
getMinOutputValue	Atomarfunktion
getPowerSource	Atomarfunktion
setdataExchangeMask	Atomarfunktion
setIOValue	Atomarfunktion
setMaxOuputValue	Atomarfunktion
setMinOuputValue	Atomarfunktion
setPowerSource	Atomarfunktion
Verhalten	
assignExceptionsConfiguration	Komfortfunktion
browseExceptions	Komfortfunktion
browseExceptionsConfiguration	Komfortfunktion
browseResponseCharacteristic	Komfortfunktion
getAlarmKey	Atomarfunktion
getAlarmMask	Atomarfunktion

getApplicationErrorMask	Atomarfunktion
getApplicationWarningMask	Atomarfunktion
getDeviceErrorMask	Atomarfunktion
getDeviceWarningMask	Atomarfunktion
getEventReaction	Atomarfunktion
getResponseCharacteristic	Atomarfunktion
getResponseMode	Atomarfunktion
setAlarmKey	Atomarfunktion
setAlarmMask	Atomarfunktion
setApplicationErrorMask	Atomarfunktion
setApplicationWarning	Atomarfunktion
setDeviceErrorMask	Atomarfunktion
setDeviceWarningMask	Atomarfunktion
setEventReaction	Atomarfunktion
setResponseCharacteristic	Atomarfunktion
setResponseMode	Atomarfunktion
Nicht Klassifizierte	
adjust	Atomarfunktion
assignConfiguration	Komfortfunktion
assignControl	Komfortfunktion
assignDeviceInformation	Komfortfunktion
assignDeviceLimits	Komfortfunktion
assignProcessInformation	Komfortfunktion
assignValueInformation	Komfortfunktion
browseConfiguration	Komfortfunktion
browseDeviceLimits	Komfortfunktion
getActuatorLimits	Atomarfunktion
getConfiguration	Atomarfunktion
getConfigurationInfo	Atomarfunktion
getControlMode	Atomarfunktion
getRedundancyConfiguration	Atomarfunktion
getSimulationValue	Atomarfunktion
getTagName	Atomarfunktion
getUserText	Atomarfunktion
protectArea	Atomarfunktion
setActuatorLimits	Atomarfunktion
setCalibrationStatus	Atomarfunktion
setConfiguration	Atomarfunktion
setConfigurationInfo	Atomarfunktion
setControlMode	Atomarfunktion
setControlType	Atomarfunktion
setDiagMode	Atomarfunktion
setFixedValues	Atomarfunktion
setRedundancyConfigurations	Atomarfunktion
setSimulationValue	Atomarfunktion
setTagName	Atomarfunktion
setToReference	Atomarfunktion

setUserText	Atomarfunktion
Parametrierung	
Fehler Warnungen	
getAlarmParameters	Atomarfunktion
getApplicationErrorParameters	Atomarfunktion
getApplicationWarningParameters	Atomarfunktion
getDeviceErrorParameters	Atomarfunktion
getDeviceWarningParameters	Atomarfunktion
getEventReactionParameters	Atomarfunktion
getFailSafeParameters	Atomarfunktion
setAlarmParameters	Atomarfunktion
setApplicationErrorParameters	Atomarfunktion
setApplicationWarningParameters	Atomarfunktion
setDeviceErrorParameters	Atomarfunktion
setDeviceWarningParameters	Atomarfunktion
setEventReactionParameters	Atomarfunktion
setFailSafeParameters	Atomarfunktion
Regler	
getAccelerationParameters	Atomarfunktion
getControlParameters	Atomarfunktion
setAccelerationParameters	Atomarfunktion
setControlParameters	Atomarfunktion
Nicht Klassifizierte	
getAuxcomPortParameters	Atomarfunktion
getCalibrationParameters	Atomarfunktion
getDevicespecificParameters	Atomarfunktion
getDisplayParameters	Atomarfunktion
getFilteringParameters	Atomarfunktion
getIOParameters	Atomarfunktion
getRedundancyParameters	Atomarfunktion
getScalingParameters	Atomarfunktion
setAuxComPortparameters	Atomarfunktion
setCalibrationParameters	Atomarfunktion
setDevicespecificParameters	Atomarfunktion
setDisplayParameters	Atomarfunktion
setFilteringParameters	Atomarfunktion
setIOParameters	Atomarfunktion
setRedundancyParameters	Atomarfunktion
setScalingParameters	Atomarfunktion
Service	
Diagnose	
browseControl	Komfortfunktion
browseMaintenance	Komfortfunktion
checkBattery	Atomarfunktion
checkConfiguration	Atomarfunktion
checkDevice	Atomarfunktion
compareApplicationCode	Atomarfunktion

functionCheck	Atomarfunktion
getAlarmInfo	Atomarfunktion
getApplicationError	Atomarfunktion
getApplicationWarning	Atomarfunktion
getControlStatus	Atomarfunktion
getDeviceError	Atomarfunktion
getDeviceWarning	Atomarfunktion
getDiagMode	Atomarfunktion
getDiagnosis	Atomarfunktion
getInitMode	Atomarfunktion
getMaintenanceData	Atomarfunktion
testDevice	Komfortfunktion
Instandhaltung	
calibrate	Komfortfunktion
getCalibrationStatus	Atomarfunktion
InitializeAuxComPort	Atomarfunktion
readbackControlWord	Atomarfunktion
readbackRatedValue	Atomarfunktion
setMaintenanceData	Atomarfunktion
startCalibration	Atomarfunktion
Instandsetzung	
assingnMaintenance	Komfortfunktion
backupDeviceData	Atomarfunktion
deleteApplication	Atomarfunktion
downloadApplicationCode	Atomarfunktion
forceApplicationData	Atomarfunktion
manipulateApplication	Komfortfunktion
manipulateDeviceState	Komfortfunktion
prepareOperation	Komfortfunktion
resetOperatingTime	Atomarfunktion
resetToDefault	Atomarfunktion
restoreDeviceData	Atomarfunktion
restoreDeviceData	Komfortfunktion
uploadApplicationCode	Atomarfunktion

Anhang D XML-Schema StrucRefArchitecture

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<!-- Version 1.2 8.402  -->
<!-- XML-Schema for the Structure Definition of DeKOS functions    -->
<!-- Date:3.9.2003 Version:1.20 Author: Heiko Meyer                   -->
<!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<xsd:schema targetNamespace="http://www.itm.tum.de/2001/adml"
xmlns="http://www.itm.tum.de/2001/adml"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
   <!--************************************************************************ -->
   <!-- STRUCTURES for DeKOSFunctions                    -->
   <!--*************************************************************************-->
   <!-- CLASS: Structures::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- XML-Schema for the Structure Definition of DeKOS functions   -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:element name="STRUCTURES">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="FILEINFO" type="fileInfoType" minOccurs="1"
maxOccurs="1"/>
            <xsd:element name="VIEW" type="viewType" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="NAME" type="xsd:string" use="required"/>
      </xsd:complexType>
   </xsd:element>
   <!--************************************************************************ -->
   <!-- Auxiliary class declarations                              -->
   <!--*************************************************************************-->
   <!-- CLASS: Fileinfo:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Global definition of the type fileinfoType, that contains the        -->
   <!-- information about some meta data of the xml File                -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="fileInfoType">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="AUTHOR" type="xsd:string" use="required"/>
            <xsd:attribute name="DATE" type="xsd:date" use="required"/>
            <xsd:attribute name="VERSION" type="xsd:float" use="required"/>
            <xsd:attribute name="FILETYPE" use="required">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                     <xsd:enumeration value="StrucDef"/>
                  </xsd:restriction>
               </xsd:simpleType>
```

```
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!-- CLASS: ViewType::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Global definition of the type structureType, that contains the        -->
   <!-- information about the Functiongroups contained in a Structure                -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="viewType">
      <xsd:sequence>
         <xsd:element name="DESCRIPTION" type="descType" minOccurs="1"
maxOccurs="1"/>
         <xsd:element name="FUNCTIONGROUP" type="functionGroupType"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="NAME" type="xsd:string" use="required"/>
   </xsd:complexType>
   <!-- CLASS: descType::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Global definition of the type descType, which holds the           -->
   <!-- description of the model items                                    -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="descType">
      <xsd:sequence>
         <xsd:element name="SPECDESC" type="xsd:string" minOccurs="0"/>
         <xsd:element name="ERGONAME" type="xsd:string" minOccurs="0"/>
         <xsd:element name="USERDESC" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="LANGUAGE" type="xsd:language"/>
   </xsd:complexType>
   <!-- CLASS: functionGroupType::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Global definition of the type functionGroupType, which contains the          -->
   <!-- the functions of a group as well as possible subgroups
-->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="functionGroupType">
      <xsd:sequence>
         <xsd:element name="DESCRIPTION" type="descType" minOccurs="1"
maxOccurs="1"/>
         <xsd:element name="FUNCTIONGROUP" type="functionGroupType" mi-
nOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="NAME" type="xsd:string" use="required"/>
      <xsd:attribute name="PRODUCTIVEFUNCTIONS" type="xsd:string"/>
      <xsd:attribute name="ATOMARFUNCTIONS" type="xsd:string"/>
      <xsd:attribute name="KOMFORTFUNCTIONS" type="xsd:string"/>
   </xsd:complexType>
</xsd:schema>
```

Anhang E XML-Dokument Sichten

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<adml:STRUCTURES NAME="Standard"
xmlns:adml="http://www.itm.tum.de/2001/adml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itm.tum.de/2001/adml StrucRefArchitec-
ture_V1.20.xsd">
   <FILEINFO AUTHOR="Heiko Meyer" DATE="2003-11-18" FILETYPE="StrucDef"
VERSION="1.2"> Standard Strukturierung</FILEINFO>
   <VIEW NAME="Gerätebeschreibung">
      <DESCRIPTION>
         <SPECDESC>Sicht enthält Klasse der Grundfunktionen zur Erstellung einer
Geräte-DDD</SPECDESC>
         <ERGONAME/>
         <USERDESC/>
      </DESCRIPTION>
      <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_2 F2_21 F2_22 F2_95 F2_130 F2_32 F2_154 F2_141 F2_48 F2_49
F2_67 F2_157 F2_85 F2_37 F2_136 F2_40 F2_42 F2_20 F2_53 F2_133 F2_55
F2_132 F2_124 F2_57 F2_58 F2_168 F2_69 F2_151 F2_94 F2_129 F2_155
F2_156 F2_150 F2_68 F2_84 F2_135 F2_5 F2_143 F2_167 F2_90 F2_91 F2_76
F2_75" KOMFORTFUNCTIONS=" " NAME="Gerätegrundfunktionen">
         <DESCRIPTION LANGUAGE="dt">
            <SPECDESC>Klasse der Grundfunktionen</SPECDESC>
            <ERGONAME/>
            <USERDESC/>
         </DESCRIPTION>
      </FUNCTIONGROUP>
   </VIEW>
   <VIEW NAME="Systembeschreibung">
      <DESCRIPTION>
         <SPECDESC>Sicht enthält Klasse der Grundfunktionen zur Erstellung einer
System-DDD</SPECDESC>
      </DESCRIPTION>
      <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_2 F2_5 F2_21 F2_130 F2_32 F2_48 F2_157 F2_37 F2_136 F2_40 F2_42
F2_20 F2_143 F2_167 F2_53 F2_133 F2_55 F2_132 F2_124 F2_57 F2_58 F2_168
F2_129 F2_135" KOMFORTFUNCTIONS=" " NAME="Systemgrundfunktionen">
         <DESCRIPTION LANGUAGE="dt">
            <SPECDESC>Klasse der Grundfunktionen</SPECDESC>
            <ERGONAME/>
            <USERDESC/>
         </DESCRIPTION>
      </FUNCTIONGROUP>
   </VIEW>
   <VIEW NAME="DeKOS_Standard">
```

```
        <DESCRIPTION>
            <SPECDESC>Sicht enthält alle Funktionen der Bibliothek</SPECDESC>
        </DESCRIPTION>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTIONS="
" KOMFORTFUNCTIONS=" " NAME="Bedienung">
            <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Klasse der Operatorfunktionen</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
            </DESCRIPTION>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_1 F2_3 F2_9 F2_25 F2_12 F2_28 F2_62 F2_37 F2_89 F2_39 F2_60
F2_42" KOMFORTFUNCTIONS="F3_1 F3_2 F3_3 F3_7 F3_8 F3_6 F3_13" NA-
ME="Anzeigen">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zum Anzeigen von allge-
meinen Informationen</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
                </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_143 F2_167" KOMFORTFUNCTIONS=" " NA-
ME="Zugangsberechtigung">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zur An- und Abmel-
dung</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
                </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_53 F2_133 F2_54 F2_123 F2_64 F2_55 F2_132 F2_124 F2_16 F2_63
F2_59 F2_92 F2_30 F2_20" KOMFORTFUNCTIONS=" " NAME="Anweisungen">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zum Betrieb</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
                </DESCRIPTION>
            </FUNCTIONGROUP>
        </FUNCTIONGROUP>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTIONS="
" KOMFORTFUNCTIONS=" " NAME="Projektierung">
            <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Klasse der Projektierungsfunktionen</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
            </DESCRIPTION>
```

```
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNC-
TIONS="F2_76 F2_130 F2_136 F2_129 F2_135 F2_75 F2_90" KOMFORTFUNC-
TIONS="F3_28 F3_16" NAME="Integration">
            <DESCRIPTION LANGUAGE="dt">
              <SPECDESC>Unterklasse der Funktionen zur Integrati-
on</SPECDESC>
              <ERGONAME/>
              <USERDESC/>
            </DESCRIPTION>
        </FUNCTIONGROUP>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="" KOMFORTFUNCTIONS="" NAME="Konfigurierung">
            <DESCRIPTION LANGUAGE="dt">
              <SPECDESC>Unterklasse der Funktionen zur Konfigurie-
rung</SPECDESC>
              <ERGONAME/>
              <USERDESC/>
            </DESCRIPTION>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_70 F2_22 F2_152 F2_148 F2_118 F2_160 F2_150 F2_155 F2_149
F2_153 F2_151 F2_69 F2_45 F2_119 F2_49 F2_154 F2_161 F2_140" KOMFORT-
FUNCTIONS="F3_27 F3_9 F3_15 F3_10" NAME="Verhalten">
              <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Unterklasse der Funtionen zum Einstellen des Gerä-
teverhaltensk</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
              </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_159 F2_100 F2_109 F2_139 F2_158 F2_13 F2_99 F2_96 F2_138" KOM-
FORTFUNCTIONS="F3_20 F3_5" NAME="Schnittstelle">
              <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Unterklasse der Funkionen zum Einstellen der
Schnittstelle</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
              </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_8 F2_79 F2_81 F2_115 F2_111 F2_19 F2_122 F2_126 F2_50 F2_134
F2_78 F2_80 F2_114 F2_61 F2_131 F2_88 F2_110 F2_18 F2_121 F2_142 F2_125
F2_66 F2_65" KOMFORTFUNCTIONS="F3_19 F3_23 F3_18 F3_24 F3_21 F3_25
F3_4 F3_12" NAME="Nicht_Klassifizierte">
              <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Nichtklassifizierter Rest der Funktio-
nen</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
```

```
                </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNC-
TIONS="F2_105 F2_103 F2_101 F2_47 F2_106 F2_104 F2_102 F2_31" KOM-
FORTFUNCTIONS=" " NAME="Einheiten_Auflösung">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zum Einstellen der Ein-
heiten, der Auflösungen etc.</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
            </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_98 F2_41 F2_56 F2_120 F2_107 F2_137 F2_97 F2_46 F2_108" KOM-
FORTFUNCTIONS=" " NAME="Messwertaufnahme">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zum Einstellen der
Messwertaufnahme</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
                </DESCRIPTION>
            </FUNCTIONGROUP>
        </FUNCTIONGROUP>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNC-
TIONS="" KOMFORTFUNCTIONS=" " NAME="Parametrierung">
            <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Unterklasse der Funktionen zur Parametrie-
rung</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
            </DESCRIPTION>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_95 F2_71 F2_34 F2_141 F2_67 F2_163 F2_85 F2_94 F2_72 F2_35
F2_156 F2_68 F2_162 F2_84" KOMFORTFUNCTIONS=" " NA-
ME="Fehler_Warnungen">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zur Einstellung des Ve-
rahaltens bei Fehlern und Warnungen</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
                </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_74 F2_117 F2_73 F2_116" KOMFORTFUNCTIONS=" " NAME="Regler">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Unterklasse der Funktionen zum Einstellen der Reg-
lerparameter</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
```

```
                </DESCRIPTION>
            </FUNCTIONGROUP>
            <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_128 F2_26 F2_147 F2_83 F2_87 F2_145 F2_113 F2_166 F2_127 F2_77
F2_146 F2_82 F2_86 F2_144 F2_112 F2_165" KOMFORTFUNCTIONS=" " NA-
ME="Nicht_Klassifizierte">
                <DESCRIPTION LANGUAGE="dt">
                    <SPECDESC>Nichtklassifizierter Rest der Funktio-
nen</SPECDESC>
                    <ERGONAME/>
                    <USERDESC/>
                </DESCRIPTION>
            </FUNCTIONGROUP>
        </FUNCTIONGROUP>
    </FUNCTIONGROUP>
    <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTIONS="
" KOMFORTFUNCTIONS=" " NAME="Service">
        <DESCRIPTION LANGUAGE="dt">
            <SPECDESC>Klasse der Servicefunktionen</SPECDESC>
            <ERGONAME/>
            <USERDESC/>
        </DESCRIPTION>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_4 F2_5 F2_6 F2_15 F2_7 F2_21 F2_24 F2_36 F2_44 F2_32 F2_48
F2_157 F2_40 F2_17 F2_33" KOMFORTFUNCTIONS="F3_11 F3_14 F3_32" NA-
ME="Diagnose">
            <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Unterklasse der Diagnosefunktionen</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
            </DESCRIPTION>
        </FUNCTIONGROUP>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_29 F2_27 F2_91 F2_51 F2_52 F2_23" KOMFORTFUNCTIONS="F3_29"
NAME="Instandhaltung">
            <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Unterklasse der Instandhaltungsfunktio-
nen</SPECDESC>
                <ERGONAME/>
                <USERDESC/>
            </DESCRIPTION>
        </FUNCTIONGROUP>
        <FUNCTIONGROUP PRODUCTIVEFUNCTIONS=" " ATOMARFUNCTI-
ONS="F2_2 F2_11 F2_168 F2_43 F2_10 F2_14 F2_58 F2_57" KOMFORTFUNC-
TIONS="F3_26 F3_33 F3_31 F3_30 F3_22" NAME="Instandsetzung">
            <DESCRIPTION LANGUAGE="dt">
                <SPECDESC>Unterklasse der Instandsetzungsfunktio-
nen</SPECDESC>
                <ERGONAME/>
```

```
                <USERDESC/>
              </DESCRIPTION>
            </FUNCTIONGROUP>
          </FUNCTIONGROUP>
        </VIEW>
</adml:STRUCTURES>
```

Anhang F XML-Schema SysRefArchitecture

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<!-- Last Changes August, 31,2003 by Heiko Meyer, itm -->
<!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<xsd:schema targetNamespace="http://www.itm.tum.de/2001/adml"
xmlns="http://www.itm.tum.de/2001/adml"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!--************************************************************************** -->
   <!-- Part 1: The DRA control functions                                  -->
   <!--***************************************************************************-->
   <!-- CLASS: Controlfunction:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type controlfunctionType, which describes the  -->
   <!-- structure of all control functions                                 -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="controlfunctionType">
      <xsd:sequence>
         <xsd:element name="DeKOSDESC" type="descType" maxOc-
curs="unbounded"/>
         <xsd:element name="BINDING" type="bindingType" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <!-- CLASS: F2function::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type f2functionType, which extends the          -->
   <!-- generic type controlfunction with the appropriate attribute group -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="f2functionType">
      <xsd:complexContent>
         <xsd:extension base="controlfunctionType">
            <xsd:sequence>
               <xsd:element name="SCRIPT" type="scriptType" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="BASICFUNCTION">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                     <xsd:enumeration value="AND"/>
                     <xsd:enumeration value="LIST"/>
                     <xsd:enumeration value="LISTALL"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
            <xsd:attribute name="DEKOSNR" type="xsd:ID" use="required"/>
            <xsd:attribute name="NAME" type="f2functionnameType"
use="required"/>
```

```
            <xsd:attribute name="FUNCTYPE">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                     <xsd:enumeration value="GET"/>
                     <xsd:enumeration value="SET"/>
                     <xsd:enumeration value="COMMAND"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <!-- CLASS: Script:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type ScriptType, which defines            -->
   <!-- the script with function-calls of a systemfunction                -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="scriptType">
      <xsd:sequence>
         <xsd:element name="STEP" maxOccurs="unbounded">
            <xsd:complexType>
               <xsd:attribute name="COMMAND" type="xsd:string"/>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="NAME" type="xsd:string"/>
      <xsd:attribute name="DESCRIPTION" type="xsd:string"/>
      <xsd:attribute name="TYPE">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="MAPPED"/>
               <xsd:enumeration value="BROADCAST"/>
               <xsd:enumeration value="ALGORITHM"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
   </xsd:complexType>
   <!-- CLASS:SubDevice:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type SubDeviceType, which describes            -->
   <!-- the underlaying devices of a subsystem                   -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="subdeviceType">
      <xsd:attribute name="NAME" type="xsd:string" use="required"/>
      <xsd:attribute name="GUID" type="xsd:string"/>
      <xsd:attribute name="DEVICE_ID" type="xsd:ID" use="required"/>
   </xsd:complexType>
   <!-- CLASS:VarDeclaration:::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the Variabletype, which defines            -->
   <!-- the variable(s) of a script                -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
```

```
   <xsd:complexType name="vardeclarationType">
      <xsd:attribute name="VAR" type="xsd:string"/>
      <xsd:attribute name="DEVICE_ID" type="xsd:string"/>
      <xsd:attribute name="DEKOSNR" type="xsd:string"/>
      <xsd:attribute name="DIRECTION">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="IN"/>
               <xsd:enumeration value="OUT"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
   </xsd:complexType>
   <!-- CLASS: F3function:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type f3functionType,which extends the             -->
   <!-- generic type controlfunction with the conn element                 -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="f3functionType">
      <xsd:complexContent>
         <xsd:extension base="controlfunctionType">
            <xsd:sequence>
               <xsd:element name="CONN" type="connType" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <!-- CLASS: StandardF3function::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type standardf3functionType,which extends the -->
   <!-- generic type f3function       with the appropriate attribute group -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="standardf3functionType">
      <xsd:complexContent>
         <xsd:extension base="f3functionType">
            <xsd:attribute name="DEKOSNR" type="xsd:ID" use="required"/>
            <xsd:attribute name="NAME" type="standardf3functionnameType"
use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <!-- CLASS: UserF3function                                  :::::::::::::::::-->
   <!-- Definition of the type userf3functionType,which extends   -->
   <!-- the type controlfunction with the appropriate attribute group      -->
   <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="userf3functionType">
      <xsd:complexContent>
         <xsd:extension base="f3functionType">
            <xsd:attribute name="DEKOSNR" type="xsd:ID" use="required"/>
            <xsd:attribute name="NAME" type="xsd:NMTOKEN" use="required"/>
```

```
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <!-- CLASS: VendorF3function                                          ::::::::::::::::::-->
   <!-- Definition of the type vendorf3functionType,which extends   -->
   <!-- the type controlfunction with the appropriate attribute group       -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="vendorf3functionType">
      <xsd:complexContent>
         <xsd:extension base="f3functionType">
            <xsd:attribute name="DEKOSNR" type="xsd:ID" use="required"/>
            <xsd:attribute name="NAME" type="xsd:NMTOKEN" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <!--***********************************************************************-->
   <!-- Part 2: The DRA interfaces                                           -->
   <!--***********************************************************************-->
   <!-- CLASS: Standardinterface::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type standardinterfaceType,which describes the-->
   <!-- element for the standard interfaces according to the DRA         -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="standardinterfaceType">
      <xsd:sequence>
         <xsd:element name="DeKOSDESC" type="descType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="NAME" type="standardinterfacenameType"/>
      <xsd:attribute name="PRODUCTIVEFUNCTIONSINTERFACE"
type="xsd:string" use="required"/>
      <xsd:attribute name="ATOMARFUNCTIONSINTERFACE" type="xsd:string"
use="required"/>
      <xsd:attribute name="KOMFORTFUNCTIONSINTERFACE" type="xsd:string"
use="required"/>
   </xsd:complexType>
   <!-- CLASS: Vendorinterface::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type vendorinterfaceType, which describes the  -->
   <!-- element for vendor-defined interfaces according to the DRA       -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="vendorinterfaceType">
      <xsd:sequence>
         <xsd:element name="DeKOSDESC" type="descType" minOccurs="0"
maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="NAME" type="xsd:NMTOKEN"/>
      <xsd:attribute name="PRODUCTIVEFUNCTIONSINTERFACE"
type="xsd:string" use="required"/>
      <xsd:attribute name="ATOMARFUNCTIONSINTERFACE" type="xsd:string"
use="required"/>
```

```
        <xsd:attribute name="KOMFORTFUNCTIONSINTERFACE" type="xsd:string"
use="required"/>
    </xsd:complexType>
    <!--*******************************************************************-->
    <!-- Part 3: The productive function                                   -->
    <!--*******************************************************************-->
    <!-- CLASS: Productivefunction:::::::::::::::::::::::::::::::::::::::::-->
    <!-- Definition of the type prodfunctionType, which describes the      -->
    <!-- structure of the productive function                              -->
    <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <xsd:complexType name="prodfunctionType">
        <xsd:complexContent>
            <xsd:extension base="controlfunctionType">
                <xsd:attribute name="DEKOSNR" type="xsd:ID" use="required"/>
                <xsd:attribute name="NAME" type="xsd:NMTOKEN" use="required"/>
                <xsd:attribute name="FUNCTYPE">
                    <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value=""/>
                            <xsd:enumeration value="GET"/>
                            <xsd:enumeration value="SET"/>
                            <xsd:enumeration value="COMMAND"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <!--*******************************************************************-->
    <!-- Part 4: The Binding for accessing Parameters                      -->
    <!--*******************************************************************-->
    <!-- CLASS: Binding:::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <!-- Definition of           -->
    <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <xsd:complexType name="bindingType">
        <xsd:sequence>
            <xsd:element name="IN" minOccurs="0">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="GENPARAM" type="genparamType" minOc-
curs="0" maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="OUT" minOccurs="0">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="GENPARAM" type="genparamType" minOc-
curs="0" maxOccurs="unbounded"/>
```

```
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="BINDTYPE" use="required">
        <xsd:simpleType>
            <xsd:restriction base="xsd:string">
                <xsd:enumeration value="EXPLICIT"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:attribute>
</xsd:complexType>
<!-- CLASS: Genparam::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<!-- Definition of                   -->
<!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<xsd:complexType name="genparamType">
    <xsd:sequence>
        <xsd:element name="DeKOSDESC" type="descType" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element name="REFTODDO" type="xsd:string" minOccurs="0"
maxOccurs="unbounded"/>
        <xsd:element name="GENPARAM" type="genparamType" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="NAME" type="genparamNameSpaceType"
use="required"/>
    <xsd:attribute name="VALUE" type="xsd:string"/>
    <xsd:attribute name="MINVALUE" type="xsd:string"/>
    <xsd:attribute name="MAXVALUE" type="xsd:string"/>
    <xsd:attribute name="TYPE" type="xsd:string"/>
</xsd:complexType>
<xsd:complexType name="mylinkType">
    <xsd:complexContent>
        <xsd:restriction base="xsd:anyType">
            <xsd:anyAttribute namespace="http://www.w3.org/1999/xlink" process-
Contents="lax"/>
        </xsd:restriction>
    </xsd:complexContent>
</xsd:complexType>
<!--**********************************************************************-->
<!-- Part 5: The CONN-Element for modelling SF-Charts              -->
<!--**********************************************************************-->
<!-- CLASS: Connector:::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<!-- Definition of the type connType, according to DRA              -->
<!-- representing the connectors between SFC-places                 -->
<!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<xsd:complexType name="connType">
    <xsd:sequence>
        <xsd:element name="PRE" minOccurs="0" maxOccurs="unbounded">
```

```
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="CONDITIONTERM" type="conditiontermType"
minOccurs="0"/>
               </xsd:sequence>
               <xsd:attribute name="PREFUNCTION" type="xsd:string"
use="required"/>
               <xsd:attribute name="UICOORDX" type="xsd:string"/>
               <xsd:attribute name="UICOORDY" type="xsd:string"/>
               <xsd:attribute name="STEPNR" type="xsd:integer"/>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="POST" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType>
               <xsd:sequence>
                  <xsd:element name="CONDITIONTERM" type="conditiontermType"
minOccurs="0"/>
               </xsd:sequence>
               <xsd:attribute name="POSTFUNCTION" type="xsd:string"
use="required"/>
               <xsd:attribute name="UICOORDX" type="xsd:string"/>
               <xsd:attribute name="UICOORDY" type="xsd:string"/>
               <xsd:attribute name="STEPNR" type="xsd:integer"/>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="CONNTYPE" use="required">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ALT"/>
               <xsd:enumeration value="PAR"/>
               <xsd:enumeration value="SEQ"/>
               <xsd:enumeration value="SIM"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:attribute>
   </xsd:complexType>
   <xsd:complexType name="conditiontermType">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="USERCOMMENT" type="xsd:string"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--********************************************************************** -->
   <!-- Part 6: Global declaration (to be used from outside with a prefix)-->
   <!--**********************************************************************-->
   <!-- CLASS: Dekosdevice:::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Global declaration of the DeKOSDEVICE element, which shall     -->
```

```
    <!-- represent the root element of a device description                -->
    <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <xsd:element name="DeKOSDEVICE">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="FILEINFO" type="fileinfoType"/>
                <xsd:element name="SUBDEVICE" type="subdeviceType" minOc-
curs="0" maxOccurs="unbounded"/>
                <xsd:element name="VARDECLARATION" type="vardeclarationType"
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="PRODFUNCTION" type="prodfunctionType"
maxOccurs="unbounded"/>
                <xsd:element name="F2FUNCTION" type="f2functionType" maxOc-
curs="165"/>
                <xsd:element name="STANDARDF3FUNCTION"
type="standardf3functionType" minOccurs="0" maxOccurs="35"/>
                <xsd:element name="USERF3FUNCTION" type="userf3functionType"
minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="VENDORF3FUNCTION"
type="vendorf3functionType" minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="STANDARDINTERFACE"
type="standardinterfaceType" maxOccurs="7"/>
                <xsd:element name="VENDORINTERFACE" type="vendorinterfaceType"
minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attributeGroup ref="deviceidentificationAttributes"/>
        </xsd:complexType>
    </xsd:element>
    <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <!-- Definition of the attributes of the root element, which comprise  -->
    <!-- the necessary device information                                  -->
    <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <xsd:attributeGroup name="deviceidentificationAttributes">
        <xsd:attribute name="VENDOR" type="xsd:string" use="required"/>
        <xsd:attribute name="DEVICENAME" type="xsd:string" use="required"/>
        <xsd:attribute name="SERNUM" type="xsd:string" use="required"/>
    </xsd:attributeGroup>
    <!--*********************************************************************** -->
    <!-- Part 7: Auxiliary class declarations                              -->
    <!--***********************************************************************-->
    <!-- CLASS: Fileinfo:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <!-- Global definition of the type fileinfoType, that contains the     -->
    <!-- information about some meta data of the xml File                  -->
    <!--:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <xsd:complexType name="fileinfoType">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="AUTHOR" type="xsd:string" use="required"/>
                <xsd:attribute name="DATE" type="xsd:date" use="required"/>
```

```
            <xsd:attribute name="VERSION" type="xsd:float" use="required"/>
            <xsd:attribute name="FILETYPE" use="required">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:NMTOKEN">
                     <xsd:enumeration value="DFL"/>
                     <xsd:enumeration value="RPE"/>
                     <xsd:enumeration value="DDD"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!-- CLASS: Dekosdesc::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Global definition of the type descType, which holds the           -->
   <!-- description of the model items                                    -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:complexType name="descType">
      <xsd:sequence>
         <xsd:element name="SPECDESC" type="xsd:string"/>
         <xsd:element name="ERGONAME" type="xsd:string"/>
         <xsd:element name="USERDESC" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="LANGUAGE" type="xsd:language" use="required"/>
   </xsd:complexType>
   <!--*****************************************************************************-->
   <!-- Part 8: Basic type declaration (DRA-namespace)                  -->
   <!--*****************************************************************************-->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <!-- Definition of the type genparamNameSpaceType, containing              -->
   <!-- an enumeration of the names of the DRA Generic Parameters   -->
   <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
   <xsd:simpleType name="genparamNameSpaceType">
      <xsd:union memberTypes="xsd:NMTOKEN genparamNameType"/>
   </xsd:simpleType>
   <xsd:simpleType name="genparamNameType">
      <xsd:restriction base="xsd:NMTOKEN">
         <xsd:enumeration value="AccessName"/>
         <xsd:enumeration value="ApplicationName"/>
         <xsd:enumeration value="ArchiveLocation"/>
         <xsd:enumeration value="CommPort"/>
         <xsd:enumeration value="CommunicationConfiguration"/>
         <xsd:enumeration value="DeviceBlock"/>
         <xsd:enumeration value="DeviceConfiguration"/>
         <xsd:enumeration value="DeviceModule"/>
         <xsd:enumeration value="Exception"/>
         <xsd:enumeration value="FunctionList"/>
         <xsd:enumeration value="GUI"/>
         <xsd:enumeration value="LinkTo"/>
```

```
        <xsd:enumeration value="ParamList"/>
        <xsd:enumeration value="Password"/>
        <xsd:enumeration value="Result"/>
        <xsd:enumeration value="Username"/>
        <xsd:enumeration value="Value"/>
        <xsd:enumeration value="AssembleValue"/>
        <xsd:enumeration value="Void"/>
        <xsd:enumeration value="ParamName"/>
        <xsd:enumeration value="ParamNameValue"/>
    </xsd:restriction>
</xsd:simpleType>
<!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<!-- Definition of the type standardinterfacenameType, containing     -->
<!-- an enumeration of the names of the DRA standardinterfaces      -->
<!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<xsd:simpleType name="standardinterfacenameType">
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="PlanningEngineer"/>
        <xsd:enumeration value="UnexpPlanningEngineer"/>
        <xsd:enumeration value="Programmer"/>
        <xsd:enumeration value="Operator"/>
        <xsd:enumeration value="Maintenance"/>
        <xsd:enumeration value="OEMService"/>
        <xsd:enumeration value="Observer"/>
    </xsd:restriction>
</xsd:simpleType>
<!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<!-- Definition of the type f2functionnameType, which describes the  -->
<!-- enumeration of the F2FUNCTION names according to the DRA  -->
<!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
<xsd:simpleType name="f2functionnameType">
    <xsd:restriction base="xsd:NMTOKEN">
        <xsd:enumeration value="browseApplication"/>
        <xsd:enumeration value="backupDeviceData"/>
        <xsd:enumeration value="restoreDeviceData"/>
        <xsd:enumeration value="browseApplicationData"/>
        <xsd:enumeration value="checkBattery"/>
        <xsd:enumeration value="checkDevice"/>
        <xsd:enumeration value="checkConfiguration"/>
        <xsd:enumeration value="compareApplicationCode"/>
        <xsd:enumeration value="adjust"/>
        <xsd:enumeration value="browseDevicespecificFunctions"/>
        <xsd:enumeration value="deleteApplication"/>
        <xsd:enumeration value="downloadApplicationCode"/>
        <xsd:enumeration value="getCertification"/>
        <xsd:enumeration value="setIOValue"/>
        <xsd:enumeration value="forceApplicationData"/>
        <xsd:enumeration value="functionCheck"/>
        <xsd:enumeration value="selectInitMode"/>
```

```
<xsd:enumeration value="getInitMode"/>
<xsd:enumeration value="setSimulationValue"/>
<xsd:enumeration value="getSimulationValue"/>
<xsd:enumeration value="gotoFailSafe"/>
<xsd:enumeration value="getAlarmInfo"/>
<xsd:enumeration value="getAlarmMask"/>
<xsd:enumeration value="InitializeAuxComPort"/>
<xsd:enumeration value="getApplicationError"/>
<xsd:enumeration value="getAuxValue"/>
<xsd:enumeration value="getCalibrationParameters"/>
<xsd:enumeration value="getCalibrationStatus"/>
<xsd:enumeration value="getComplianceInfo"/>
<xsd:enumeration value="startCalibration"/>
<xsd:enumeration value="stopApplication"/>
<xsd:enumeration value="getDecPointPos"/>
<xsd:enumeration value="getDeviceError"/>
<xsd:enumeration value="getDiagMode"/>
<xsd:enumeration value="getApplicationWarningParameters"/>
<xsd:enumeration value="setApplicationWarningParameters"/>
<xsd:enumeration value="getApplicationWarning"/>
<xsd:enumeration value="getIdentification"/>
<xsd:enumeration value="getIOValue"/>
<xsd:enumeration value="getMaintenanceData"/>
<xsd:enumeration value="getMeasurementOffsetValue"/>
<xsd:enumeration value="getOperatingTime"/>
<xsd:enumeration value="uploadApplicationCode"/>
<xsd:enumeration value="getControlStatus"/>
<xsd:enumeration value="getResponseMode"/>
<xsd:enumeration value="getSensorLimits"/>
<xsd:enumeration value="setDecPointPos"/>
<xsd:enumeration value="getDeviceWarning"/>
<xsd:enumeration value="protectArea"/>
<xsd:enumeration value="readbackControlWord"/>
<xsd:enumeration value="readbackRatedValue"/>
<xsd:enumeration value="resetAlarm"/>
<xsd:enumeration value="resetApplication"/>
<xsd:enumeration value="resetDevice"/>
<xsd:enumeration value="resetMeasurementOffsetValue"/>
<xsd:enumeration value="resetOperatingTime"/>
<xsd:enumeration value="resetToDefault"/>
<xsd:enumeration value="setOperatingMode"/>
<xsd:enumeration value="getOperatingMode"/>
<xsd:enumeration value="selectControlType"/>
<xsd:enumeration value="setControlType"/>
<xsd:enumeration value="getControlType"/>
<xsd:enumeration value="selectParametersSet"/>
<xsd:enumeration value="setActuatorLimits"/>
<xsd:enumeration value="getActuatorLimits"/>
<xsd:enumeration value="getDeviceWarningParameters"/>
```

```
<xsd:enumeration value="setDeviceWarningParameters"/>
<xsd:enumeration value="setAlarmKey"/>
<xsd:enumeration value="getAlarmKey"/>
<xsd:enumeration value="getApplicationErrorParameters"/>
<xsd:enumeration value="setApplicationErrorParameters"/>
<xsd:enumeration value="setBrakeParameters"/>
<xsd:enumeration value="getBrakeParameters"/>
<xsd:enumeration value="setCommunicationParameters"/>
<xsd:enumeration value="getCommunicationParameters"/>
<xsd:enumeration value="setCalibrationParameters"/>
<xsd:enumeration value="setConfiguration"/>
<xsd:enumeration value="getConfiguration"/>
<xsd:enumeration value="setConfigurationInfo"/>
<xsd:enumeration value="getConfigurationInfo"/>
<xsd:enumeration value="setDisplayParameters"/>
<xsd:enumeration value="getDisplayParameters"/>
<xsd:enumeration value="setFailSafeParameters"/>
<xsd:enumeration value="getFailSafeParameters"/>
<xsd:enumeration value="setFilteringParameters"/>
<xsd:enumeration value="getFilteringParameters"/>
<xsd:enumeration value="setFixedValues"/>
<xsd:enumeration value="getFixedValues"/>
<xsd:enumeration value="setIdentification"/>
<xsd:enumeration value="setMaintenanceData"/>
<xsd:enumeration value="startApplication"/>
<xsd:enumeration value="setAlarmParameters"/>
<xsd:enumeration value="getAlarmParameters"/>
<xsd:enumeration value="setMinOutputValue"/>
<xsd:enumeration value="setMaxMeasurementValue"/>
<xsd:enumeration value="getMaxMeasurementValue"/>
<xsd:enumeration value="setMaxOuputValue"/>
<xsd:enumeration value="getMaxOuputValue"/>
<xsd:enumeration value="setMeasurementDimension"/>
<xsd:enumeration value="getMeasurementDimension"/>
<xsd:enumeration value="setMeasurementResolution"/>
<xsd:enumeration value="getMeasurementResolution"/>
<xsd:enumeration value="setMinMeasurementValue"/>
<xsd:enumeration value="getMinMeasurementValue"/>
<xsd:enumeration value="getMinOutputValue"/>
<xsd:enumeration value="setRedundancyConfiguration"/>
<xsd:enumeration value="getRedundancyConfiguration"/>
<xsd:enumeration value="setRedundancyParameters"/>
<xsd:enumeration value="getRedundancyParameters"/>
<xsd:enumeration value="setControlMode"/>
<xsd:enumeration value="getControlMode"/>
<xsd:enumeration value="setControlParameters"/>
<xsd:enumeration value="getControlParameters"/>
<xsd:enumeration value="setResponseCharacteristic"/>
<xsd:enumeration value="getResponseCharacteristic"/>
```

```
<xsd:enumeration value="setSensorLimits"/>
<xsd:enumeration value="setTagName"/>
<xsd:enumeration value="getTagName"/>
<xsd:enumeration value="setUnit"/>
<xsd:enumeration value="getUnit"/>
<xsd:enumeration value="setUserText"/>
<xsd:enumeration value="getUserText"/>
<xsd:enumeration value="setAuxComPortParameters"/>
<xsd:enumeration value="getAuxComPortParameters"/>
<xsd:enumeration value="setCountry"/>
<xsd:enumeration value="getCountry"/>
<xsd:enumeration value="setDiagMode"/>
<xsd:enumeration value="setCalibrationStatus"/>
<xsd:enumeration value="setLanguage"/>
<xsd:enumeration value="getLanguage"/>
<xsd:enumeration value="setMeasurementOffsetValue"/>
<xsd:enumeration value="setPowerSource"/>
<xsd:enumeration value="getPowerSource"/>
<xsd:enumeration value="setResponseMode"/>
<xsd:enumeration value="getDeviceErrorParameters"/>
<xsd:enumeration value="setDeviceErrorParameters"/>
<xsd:enumeration value="setToReference"/>
<xsd:enumeration value="login"/>
<xsd:enumeration value="logout"/>
<xsd:enumeration value="setIOParameters"/>
<xsd:enumeration value="getIOParameters"/>
<xsd:enumeration value="setDevicespecificParameters"/>
<xsd:enumeration value="getDevicespecificParameters"/>
<xsd:enumeration value="getApplicationWarningMask"/>
<xsd:enumeration value="setApplicationWarningMask"/>
<xsd:enumeration value="setAlarmMask"/>
<xsd:enumeration value="getApplicationErrorMask"/>
<xsd:enumeration value="setApplicationErrorMask"/>
<xsd:enumeration value="setHWErrorParameters"/>
<xsd:enumeration value="getDiagnosis"/>
<xsd:enumeration value="resetAllExceptions"/>
<xsd:enumeration value="resetDeviceError"/>
<xsd:enumeration value="resetDeviceWarning"/>
<xsd:enumeration value="resetApplicationError"/>
<xsd:enumeration value="resetApplicationWarning"/>
<xsd:enumeration value="setDataExchangeMask"/>
<xsd:enumeration value="getDataExchangeMask"/>
<xsd:enumeration value="setEventReaction"/>
<xsd:enumeration value="getEventReaction"/>
<xsd:enumeration value="setEventReactionParameters"/>
<xsd:enumeration value="getEventReactionParameters"/>
<xsd:enumeration value="setAccelerationParameters"/>
<xsd:enumeration value="getAccelerationParameters"/>
<xsd:enumeration value="getDeviceErrorMask"/>
```

```
            <xsd:enumeration value="setDeviceErrorMask"/>
            <xsd:enumeration value="getDeviceWarningMask"/>
            <xsd:enumeration value="setDeviceWarningMask"/>
            <xsd:enumeration value="setScalingParameters"/>
            <xsd:enumeration value="getScalingParameters"/>
        </xsd:restriction>
    </xsd:simpleType>
    <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <!-- Definition of the type standardf3functionnameType, which          -->
    <!-- containing an enumeration of the STANDARDF3FUNCTION       -->
    <!-- names according to the DRA                                         -->
    <!--::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::-->
    <xsd:simpleType name="standardf3functionnameType">
        <xsd:restriction base="xsd:NMTOKEN">
            <xsd:enumeration value="browseDeviceIdentification"/>
            <xsd:enumeration value="browseDeviceInformation"/>
            <xsd:enumeration value="browseMotionSets"/>
            <xsd:enumeration value="browseConfiguration"/>
            <xsd:enumeration value="browseIOConfiguration"/>
            <xsd:enumeration value="browseProcessInformation"/>
            <xsd:enumeration value="browseOperatingState"/>
            <xsd:enumeration value="browseOperatingValues"/>
            <xsd:enumeration value="browseExceptions"/>
            <xsd:enumeration value="browseResponseCharacteristic"/>
            <xsd:enumeration value="browseControl"/>
            <xsd:enumeration value="browseDeviceLimits"/>
            <xsd:enumeration value="browseValueInformation"/>
            <xsd:enumeration value="browseMaintenance"/>
            <xsd:enumeration value="browseExceptionsConfiguration"/>
            <xsd:enumeration value="browseCommunication"/>
            <xsd:enumeration value="assignDeviceInformation"/>
            <xsd:enumeration value="assignConfiguration"/>
            <xsd:enumeration value="assignIOConfiguration"/>
            <xsd:enumeration value="assignProcessInformation"/>
            <xsd:enumeration value="assignControl"/>
            <xsd:enumeration value="assignDeviceLimits"/>
            <xsd:enumeration value="assignValueInformation"/>
            <xsd:enumeration value="assignMaintenance"/>
            <xsd:enumeration value="assignExceptionsConfiguration"/>
            <xsd:enumeration value="assingCommunication"/>
            <xsd:enumeration value="calibrate"/>
            <xsd:enumeration value="prepareOperation"/>
            <xsd:enumeration value="manipulateDeviceState"/>
            <xsd:enumeration value="testDevice"/>
            <xsd:enumeration value="manipulateApplication"/>
            <xsd:enumeration value="restoreDeviceData"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

Anhang G DDD des Beispielsystems

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<adml:DeKOSDEVICE DEVICENAME="Partialsystem Pumpstation" SER-
NUM="PS17" VENDOR="itm" xmlns:adml="http://www.itm.tum.de/2001/adml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.itm.tum.de/2001/adml DRA_V1.20.xsd">
   <FILEINFO AUTHOR="Heiko Meyer" DATE="2003-12-14" FILETYPE="DDD"
VERSION="1.4">Systembeschreibung des Partialsystems Pumpstation</FILEINFO>
   <SUBDEVICE NAME="TPE Pumpe" GUID="59CC0C20-679B-11D2-88BD-
0800361A1803" DEVICE_ID="D1"/>
   <SUBDEVICE NAME="Auma Matic" GUID="5E162AD9-D430-4B1F-80DB-
794BA0AD3C18" DEVICE_ID="D2"/>
   <SUBDEVICE NAME="Promag 33" GUID="FBF6CA44-94F5-44A2-A4B5-
3C122B8A89B7" DEVICE_ID="D3"/>
   <VARDECLARATION VAR="startPump" DEKOSNR="PF_1" DEVICE_ID="D1"
DIRECTION="OUT"/>
   <VARDECLARATION VAR="openValve" DEKOSNR="PF_1" DEVICE_ID="D2"
DIRECTION="OUT"/>
   <VARDECLARATION VAR="getFlowRate" DEKOSNR="PF_1" DEVICE_ID="D3"
DIRECTION="IN"/>
   <PRODFUNCTION DEKOSNR="PF_1" NAME="pump" FUNC-
TYPE="COMMAND">
      <DeKOSDESC LANGUAGE="dt">
         <SPECDESC/>
         <ERGONAME>Fluid pumpen</ERGONAME>
         <USERDESC>Pumpt entweder kontinuierlich oder eine zuvor festgelegte
Menge des Fluids.</USERDESC>
      </DeKOSDESC>
   </PRODFUNCTION>
   <F2FUNCTION DEKOSNR="F2_2" FUNCTYPE="COMMAND"
NAME="backupDeviceData" BASICFUNCTION="AND">
      <DeKOSDESC LANGUAGE="dt">
         <SPECDESC>Gerätedaten, -Parameter, Speicherblöcke etc. werden auf
Datenträger gesichert.</SPECDESC>
         <ERGONAME>Gerätedaten sichern</ERGONAME>
         <USERDESC>user</USERDESC>
      </DeKOSDESC>
   </F2FUNCTION>
   <F2FUNCTION DEKOSNR="F2_5" FUNCTYPE="COMMAND"
NAME="checkDevice" BASICFUNCTION="LIST">
      <DeKOSDESC LANGUAGE="dt">
         <SPECDESC>Leitet eine Überprüfung der Feldgeräte-Hardware ein, führt
also eine Art Selbstdiagnose bezüglich Prozessor, Speicher etc. aus.</SPECDESC>
         <ERGONAME>Gerätehardware prüfen</ERGONAME>
         <USERDESC>user</USERDESC>
      </DeKOSDESC>
```

```
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_20" FUNCTYPE="COMMAND"
NAME="gotoFailSafe" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Schaltet die Ausgänge der Feldgeräte in den sicheren Zu-
stand.</SPECDESC>
            <ERGONAME>Fail Safe Zustand aktivieren</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_21" FUNCTYPE="GET" NAME="getAlarmInfo"
BASICFUNCTION="LIST">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest Informationen zu einem Alarm aus den Feldgeräten aus
(Textuelle Informationen, Status_quittiert/unquittiert, anstehend/nicht anstehend, ak-
tiv/inaktiv). Dieser wird vom Prozess ausgelöst (Prozessgrösse).</SPECDESC>
            <ERGONAME>Alarminformationen lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_32" FUNCTYPE="GET" NAME="getDeviceError"
BASICFUNCTION="LIST">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest Gerätefehler aus den Feldgeräten aus.</SPECDESC>
            <ERGONAME>Gerätefehler lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_37" FUNCTYPE="GET"
NAME="getIdentification" BASICFUNCTION="LISTALL">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest Identifikationsdaten aus den Feldgeräten
aus.</SPECDESC>
            <ERGONAME>Geräteidentifikation lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_40" FUNCTYPE="GET"
NAME="getMaintenanceData" BASICFUNCTION="LIST">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest Instandhaltungsdaten aus den Feldgeräten in das
Tool.</SPECDESC>
            <ERGONAME>Instandhaltungsdaten lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_42" FUNCTYPE="GET"
NAME="getOperatingTime" BASICFUNCTION="LISTALL">
        <DeKOSDESC LANGUAGE="dt">
```

```
            <SPECDESC>Liest Daten aus den Feldgeräten, die die Betriebsdauer
betreffen.</SPECDESC>
            <ERGONAME>Betriebsdauer lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_48" FUNCTYPE="GET"
NAME="getDeviceWarning" BASICFUNCTION="LIST">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest eine Warnmeldung aus den Feldgeräten aus, (Textuelle
Informationen, Status_ quittiert/unquittiert, anstehend/nicht anstehend, ak-
tiv/inaktiv).</SPECDESC>
            <ERGONAME>Gerätewarnung lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_53" FUNCTYPE="COMMAND"
NAME="resetAlarm" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Quittiert Alarmmeldungen.</SPECDESC>
            <ERGONAME>Alarm quittieren</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_55" FUNCTYPE="COMMAND"
NAME="resetDevice" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Warmstart der Feldgeräte einleiten.</SPECDESC>
            <ERGONAME>Warmstart</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_57" FUNCTYPE="SET"
NAME="resetOperatingTime" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Setzt die Betriebsstundenzähler zurück.</SPECDESC>
            <ERGONAME>Betriebsstundenzähler zurücksetzen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_58" FUNCTYPE="SET" NAME="resetToDefault"
BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Rücksetzen der Geräte auf die Werkseinstellungen. Falls ein-
zelne Komponenten zurückgesetzt werden können, ist die Angabe dieser mög-
lich.</SPECDESC>
            <ERGONAME>Werkeinstellungen setzen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
```

```
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_124" FUNCTYPE="COMMAND"
NAME="resetDeviceWarning" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Quittiert Gerätewarnungen.</SPECDESC>
            <ERGONAME>Gerätewarnung quittieren</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_129" FUNCTYPE="SET" NAME="setCountry"
BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Stellt das Land ein, dessen Masseinheiten etc. gelten sol-
len.</SPECDESC>
            <ERGONAME>Ländereinstellung setzen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_130" FUNCTYPE="GET" NAME="getCountry"
BASICFUNCTION="LISTALL">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest die aktuelle Landeseinstellung aus.</SPECDESC>
            <ERGONAME>Landeseinstellung lesen</ERGONAME>
        <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_132" FUNCTYPE="COMMAND"
NAME="resetDeviceError" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Quittiert Gerätefehler.</SPECDESC>
            <ERGONAME>Gerätefehler quittieren</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_133" FUNCTYPE="COMMAND"
NAME="resetAllExceptions" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Quittiert alle Alarme, Warnungen, Hard- und Softwarefehler
und -warnungen.</SPECDESC>
            <ERGONAME>Alle Alarme und Warnungen quittieren</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_135" FUNCTYPE="SET" NAME="setLanguage"
BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Stellt die Spracheinstellungen für die Feldgeräte
ein.</SPECDESC>
            <ERGONAME>Spracheinstellung setzen</ERGONAME>
```

```
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_136" FUNCTYPE="GET" NAME="getLanguage"
BASICFUNCTION="LISTALL">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest die Spracheinstellung für die Feldgeräte.</SPECDESC>
            <ERGONAME>Spracheinstellung lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_143" FUNCTYPE="COMMAND" NAME="login"
BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Am Partialsystem/Feldgeräten anmelden.</SPECDESC>
            <ERGONAME>Anmelden</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_157" FUNCTYPE="GET"
NAME="getDiagnosis">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Liest Diagnosemeldungen des Partialsystems
aus.</SPECDESC>
            <ERGONAME>Diagnosemeldung lesen</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
        <SCRIPT NAME="Dichtigkeitsprüfung" DESCRIPTION="Selbstdiagnose durch
Pumpen gegen geschlossenes Ventil zur Kontrolle der Dichtung" TY-
PE="ALGORITHM">
            <STEP COMMAND="Subsystem.getDiagnosis()"/>
            <STEP COMMAND="openValve() = false"/>
            <STEP COMMAND="startPump() = true"/>
            <STEP COMMAND="resetTimer"/>
            <STEP COMMAND="startTimer"/>
            <STEP COMMAND="while Timer != 10 do value = getFlowRate()"/>
            <STEP COMMAND="if value > 0 then returnvalue = 1 else returnvalue =
0"/>
            <STEP COMMAND="startPump() = false"/>
            <STEP COMMAND="stopTimer"/>
        </SCRIPT>
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_167" FUNCTYPE="COMMAND" NAME="logout"
BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Am Partialsystem/Feldgeräten abmelden.</SPECDESC>
            <ERGONAME>Abmelden</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
```

```
    </F2FUNCTION>
    <F2FUNCTION DEKOSNR="F2_168" FUNCTYPE="COMMAND"
NAME="restoreDeviceData" BASICFUNCTION="AND">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Gerätedaten, -Parameter, Speicherblöcke etc. werden in die
Geräte zurück geschrieben.</SPECDESC>
            <ERGONAME>Gerätedaten zurückschreiben</ERGONAME>
            <USERDESC>user</USERDESC>
        </DeKOSDESC>
    </F2FUNCTION>
    <STANDARDINTERFACE ATOMARFUNCTIONSINTERFACE="F2_1 F2_2 F2_3
F2_4 F2_5 F2_6 F2_7 F2_9 F2_10 F2_11 F2_12 F2_14 F2_15 F2_16 F2_17 F2_18
F2_19 F2_20 F2_21 F2_22 F2_23 F2_24 F2_25 F2_26 F2_27 F2_28 F2_29 F2_30
F2_31 F2_32 F2_33 F2_34 F2_35 F2_36 F2_37 F2_38 F2_39 F2_40 F2_41 F2_42
F2_44 F2_45 F2_46 F2_47 F2_48 F2_49 F2_50 F2_51 F2_52 F2_53 F2_54 F2_55
F2_56 F2_57 F2_58 F2_59 F2_60 F2_61 F2_62 F2_63 F2_64 F2_65 F2_66 F2_67
F2_68 F2_69 F2_70 F2_71 F2_72 F2_73 F2_74 F2_75 F2_76 F2_77 F2_78 F2_79
F2_80 F2_81 F2_82 F2_83 F2_84 F2_85 F2_86 F2_87 F2_88 F2_89 F2_91 F2_92
F2_93 F2_94 F2_95 F2_96 F2_97 F2_98 F2_99 F2_100 F2_101 F2_102 F2_103
F2_104 F2_105 F2_106 F2_107 F2_108 F2_109 F2_110 F2_111 F2_112 F2_113
F2_114 F2_115 F2_116 F2_117 F2_118 F2_119 F2_120 F2_121 F2_122 F2_123
F2_124 F2_125 F2_126 F2_127 F2_128 F2_129 F2_130 F2_131 F2_132 F2_133
F2_134 F2_135 F2_136 F2_137 F2_138 F2_139 F2_140 F2_141 F2_142 F2_143
F2_144 F2_145 F2_146 F2_147 F2_148 F2_149 F2_150 F2_151 F2_152 F2_153
F2_154 F2_155 F2_156 F2_157 F2_158 F2_159 F2_160 F2_161 F2_162 F2_163
F2_164 F2_165 F2_166 F2_167 F2_168" KOMFORTFUNCTIONSINTERFA-
CE="F3_1 F3_2 F3_3 F3_4 F3_5 F3_6 F3_7 F3_8 F3_9 F3_10 F3_11 F3_12 F3_13
F3_14 F3_15 F3_16 F3_18 F3_19 F3_20 F3_21 F3_22 F3_23 F3_24 F3_25 F3_26
F3_27 F3_28 F3_29 F3_30 F3_31 F3_32 F3_33" PRODUCTIVEFUNCTIONSIN-
TERFACE="" NAME="PlanningEngineer">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Dieses Interface liefert Funktionen für den erfahrenen In-
betriebnehmer.</SPECDESC>
            <ERGONAME>Erfahrener Inbetriebnehmer</ERGONAME>
            <USERDESC/>
        </DeKOSDESC>
    </STANDARDINTERFACE>
    <STANDARDINTERFACE ATOMARFUNCTIONSINTERFACE="F2_1 F2_2 F2_3
F2_4 F2_5 F2_8 F2_9 F2_10 F2_11 F2_12 F2_13 F2_14 F2_15 F2_17 F2_18
F2_19 F2_20 F2_21 F2_22 F2_23 F2_24 F2_25 F2_26 F2_27 F2_28 F2_29 F2_30
F2_31 F2_32 F2_33 F2_35 F2_34 F2_36 F2_37 F2_38 F2_39 F2_40 F2_41 F2_42
F2_43 F2_44 F2_45 F2_46 F2_47 F2_48 F2_49 F2_51 F2_52 F2_53 F2_54 F2_55
F2_56 F2_57 F2_58 F2_59 F2_60 F2_62 F2_63 F2_64 F2_66 F2_67 F2_68 F2_70
F2_71 F2_72 F2_74 F2_76 F2_77 F2_79 F2_81 F2_82 F2_83 F2_85 F2_86 F2_87
F2_88 F2_89 F2_91 F2_92 F2_94 F2_95 F2_96 F2_97 F2_98 F2_99 F2_100
F2_101 F2_102 F2_103 F2_104 F2_105 F2_106 F2_107 F2_108 F2_109 F2_111
F2_113 F2_115 F2_116 F2_117 F2_118 F2_119 F2_120 F2_121 F2_122 F2_123
F2_124 F2_125 F2_126 F2_127 F2_128 F2_130 F2_131 F2_132 F2_133 F2_134
F2_136 F2_137 F2_138 F2_139 F2_140 F2_141 F2_142 F2_143 F2_144 F2_145
```

```
F2_146 F2_147 F2_148 F2_149 F2_150 F2_151 F2_152 F2_153 F2_154 F2_155
F2_156 F2_157 F2_164 F2_167 F2_168" KOMFORTFUNCTIONSINTERFA-
CE="F3_1 F3_2 F3_3 F3_4 F3_5 F3_6 F3_7 F3_8 F3_9 F3_10 F3_11 F3_12 F3_13
F3_14 F3_15 F3_16 F3_21 F3_22 F3_24 F3_25 F3_26 F3_29 F3_30 F3_31 F3_32"
PRODUCTIVEFUNCTIONSINTERFACE="" NAME="Maintenance">
        <DeKOSDESC LANGUAGE="dt">
            <SPECDESC>Dieses Interface liefert Instandhaltungsfunktio-
nen.</SPECDESC>
            <ERGONAME>Instandhaltung</ERGONAME>
            <USERDESC/>
        </DeKOSDESC>
    </STANDARDINTERFACE>
</adml:DeKOSDEVICE>
```