Funktionaler Anwendungsentwurf verteilter Automatisierungssysteme

– Anwendung von Entwurfsmustern in der Automatisierungstechnik –

Von der Fakultät für Maschinenbau
der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg
zur Erlangung des akademischen Grades eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

Dissertation
vorgelegt von

M. Sc. Karin Eckert

aus Leonberg

Hamburg 2015

Tag der mündlichen Prüfung: 8. Mai 2015

Gutachter: Prof. Dr.-Ing. Alexander Fay
Prof. Dr.-Ing. Birgit Vogel-Heuser

Vorwort der Autorin

So eine Arbeit wird eigentlich nie fertig, man muß sie für fertig erklären, wenn man nach Zeit und Umständen das Mögliche getan hat.

Johann Wolfgang von Goethe

Die vorliegende Arbeit entstand während meiner Zeit als Wissenschaftliche Mitarbeiterin an der Professur für Automatisierungstechnik der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg von Oktober 2010 bis Dezember 2013.

Ich möchte im Nachfolgenden allen Menschen meinen Dank aussprechen, die maßgeblich zum Entstehen dieser Arbeit beigetragen haben.

Mein besonderer Dank gilt Herrn Professor Dr.-Ing. Alexander Fay für die Betreuung meiner Arbeit sowie die Mitarbeit an seiner Professur. Er ermöglichte mir darüber hinaus interessante Lehrtätigkeiten sowie Forschungsprojekte. In zahlreichen Gesprächen und Diskussionen gab er mir vielfältige Anregungen und Impulse, welche maßgeblich zum Gelingen dieser Arbeit beigetragen haben.

Des Weiteren danke ich Frau Professor Dr.-Ing. Birgit Vogel-Heuser für die Anfertigung des Zweitgutachtens sowie ihr Interesse an meiner Arbeit. Ebenso danke ich Herrn Professor Dr.-Ing. Martin Meywerk für die Übernahme des Prüfungsvorsitzes.

An dieser Stelle möchte ich mich auch bei meinen ehemaligen Kollegen für die gemeinsame Zeit am Institut bedanken. Die zahlreichen Diskussionen und auch die respektvolle und kollegiale Zusammenarbeit werden mir in positiver Erinnerung bleiben. Insbesondere möchte ich mich auch bei Herrn Dr.-Ing. Frank Schumacher, Herrn Dr.-Ing. Sebastian Schreiber und Herrn Ireneus Wior bedanken, die mir bei der kritischen Durchsicht dieser Arbeit zahlreiche Anmerkungen und Ratschläge gaben.

Nicht zuletzt möchte ich mich bei meiner Familie bedanken, die mich in meiner Arbeit bestärkt hat und mir Ihr Verständnis und Ihre Unterstützung in den vergangenen Jahren besonders zuteilwerden ließ. Insbesondere meinen Eltern sowie meinen Schwestern möchte ich danken, dass sie stets an mich geglaubt haben. Dabei gebührt ein besonderer Dank meinem Vater, Ewald Eckert, der mich auch in schwierigen Zeiten immer unterstützt hat und bei der Durchsicht meiner Arbeit zahlreiche Ratschläge gegeben hat. Des Weiteren danke ich meiner Schwester, Sarah Eckert, für ihre zahlreichen Anregungen bei der Durchsicht meiner Arbeit.

Magstadt, im Mai 2015 Karin Eckert

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

A

AIS . . . . . Lehrstuhl für Automatisierung und Informationssysteme der Technischen Universität München
AT Hardware . . . . . automatisierungstechnische Hardware
AT Software . . . . . automatisierungstechnische Software
AUTOSAR . . . . . AUTomotive Open System ARchitecture

B

BWL . . . . . Betriebswirtschaftslehre

C

CPS . . . . . Cyber-Physical Systems

F

FB . . . . . Funktionsblock
FBS . . . . . Funktionsbausteinsprache
FMS . . . . . Flexible Manufacturing System

G

GoF . . . . . Gang of Four
GRAFCET . . . . . GRAphe Fonctionnel de Commande Etape Transition

I

IFAT . . . . . Institut für Automatisierungstechnik der Otto-von-Guericke-Universität Magdeburg
IT . . . . . Informationstechnik

N

nfA . . . . . nicht-funktionale Anforderung

O

OMG . . . . . Object Management Group

P

PEARL . . . . . Process and Experiment Automation Realtime Language
PLS . . . . . Prozessleitsystem
PLT . . . . . Prozessleittechnik
POU . . . . . Program Organization Unit

R

R&I-Fließbild . . . . . Rohrleitungs- und Intrumenten-Fließbild

S

SOCRADES . . . . . Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded deviceS

SysML . . . . . Systems Modeling Language

U

UML-PA . . . . . Unified Modelling Language for Process Automation

UML . . . . . Unified Modelling Language

1 Einleitung

In dieser Einleitung wird zunächst die Motivation für die Erstellung der Arbeit aufgezeigt, und daraus werden die Herausforderungen des Engineerings abgeleitet. Abschließend erfolgen die Formulierung der Zielsetzung sowie die Darstellung und Erläuterung des Aufbaus der Arbeit.

1.1 Motivation

Die zunehmende Komplexität von Automatisierungssystemen [Automation 2012]@ hat zur Folge, dass diese anfälliger für Störungen und Fehler sind. Die Komplexität ist dadurch begründet, dass die gewünschte Funktionalität von Automatisierungssystemen sowie die Anforderungen, wie beispielsweise höchste Qualität mit hoher Zuverlässigkeit und funktionaler Sicherheit zu geringstmöglichen Kosten und in kürzester Zeit, an solche Systeme stetig steigen. Hieraus resultiert die Forderung, automatisierte Systeme gesamtheitlich optimal zu betreiben, und zwar durch gezielte Verteilung der Funktionalität im System. Für die Steuerung und Regelung technischer Prozesse müssen Informationen von verschiedenen Komponenten, wie beispielsweise Sensoren, verknüpft, gemeinsam verarbeitet und an verschiedene Informationsempfänger, wie beispielsweise Aktoren, verteilt werden. Dies führt zu einer Zunahme der informationstechnischen Kopplung zwischen zuvor getrennten Regelungs- und Steuerungsaufgaben [vgl. EIT Bericht 2010, S. 5 f.]. Bedingt durch die oftmals räumliche Ausdehnung der automatisierten Anlagen, führt dies zu verteilten Automatisierungssystemen, in denen verschiedene Komponenten Funktionalität ausführen und miteinander über Kommunikationsnetze verbunden sind [vgl. EIT Bericht 2010, S. 5 f.]. Diese verteilten Automatisierungssysteme werden aus unterschiedlichen Komponenten zusammengesetzt, die jeweils aus Hard- und Software bestehen und zumeist hinsichtlich der Funktionen, die diese Komponenten ausführen können, anpassbar sind. Im Hinblick auf Art und Anzahl solcher Komponenten sowie deren Eigenschaften und Verknüpfungen sind solche verteilten Automatisierungssysteme komplex. Der Entwurf dieser Systeme, bei dem insbesondere die Verteilung der Funktionalität auf die Komponenten und deren Kommunikation zu entscheiden sind, ist somit ein komplexes Optimierungsproblem, das gelöst werden muss, da verteilte Strukturen in jüngster Vergangenheit zunehmend an Bedeutung gewinnen (siehe Industrie 4.0 [Industrie 4.0]@) [vgl. Fay 2005, S. 205 f.] [vgl. Dechema 2009, S. 34 f.].

Der Entwurf von Steuerungssoftware und deren Verteilung auf die Hardware ist ein wesentlicher Bestandteil des Engineering-Prozesses von Automatisierungssystemen [Vogel-Heuser et al. 2013]. Die für das Engineering notwendigen Arbeitsschritte umfassen die Bereiche von der Anforderungserhebung bis hin zur Wartung und Optimierung [vgl. Vogel-Heuser et al. 2013, S. 7]. Der Entwurf ist somit ein wichtiger Bestandteil des Engineerings und kann in das Phasenmodell des Engineerings eingeordnet werden (siehe Abbildung 1.1 auf der nächsten Seite).

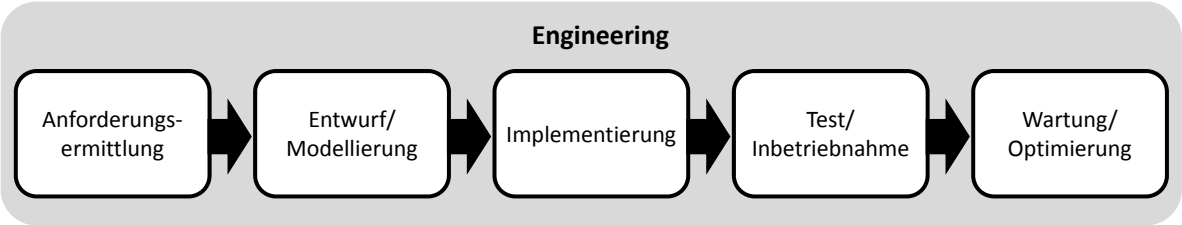

Abbildung 1.1: Phasen des Engineerings

Für das Engineering von Systemen existiert eine Vielzahl an Vorgehensmodellen, die im Bereich des Maschinenbaus oder der Softwaretechnik Anwendung finden. Ein wesentlicher Aspekt, den generische Vorgehensmodelle erfüllen müssen, ist, dass diese auf den jeweiligen Anwendungsfall sowie die zugehörige Domäne angepasst werden. Klassische Vorgehensmodelle des Maschinenbaus sowie der Prozessleittechnik (PLT), wie beispielsweise NA 35 [NA 35 2003][#], können aufgrund ihres Detaillierungsgrads sowie von fehlenden Iterationen (Rücksprünge) und Schnittstellen zwischen den Phasen an die Problemstellungen und Herausforderungen verteilter Automatisierungssysteme schwierig angepasst werden. Vereinzelt existieren Ansätze, welche die Entwicklung von verteilten Automatisierungssystemen fördern. Diese Ansätze geben aber keine explizite Unterstützung zur Erfassung von nicht-funktionalen Anforderungen (nfAs). Wie zuvor beschrieben, besteht ein Defizit bezüglich eines Vorgehensmodells, das den Entwurf von Automatisierungssystemen unterstützt, mit besonderem Fokus auf Verteilung der Funktionalität und unter Berücksichtigung von funktionalen und nicht-funktionalen Anforderungen. Zumeist werden die Anforderungen an ein Automatisierungssystem nicht technologisch, sondern funktional beschrieben. Dies bedeutet, dass technologiebedingte Einschränkungen, wie beispielsweise Kommunikationstechnologien, vermieden werden und das Augenmerk auf den funktionalen Anforderungen eines Systems liegt.

Existierende Methoden stellen für gegebene funktionale Anforderungen verschiedene Lösungen für Teilprobleme des Entwurfs (Auswahl von Komponenten oder Entwurf von Programmabläufen) bereit. Diese Methoden und Verfahren sind zumeist mathematisch komplex (Algorithmen) und somit nur bedingt auf große Systeme anwendbar. Außerdem werden oftmals vollständig spezifizierte Anforderungen benötigt, was für große Systeme nicht erfüllbar ist. Der Fachbereich Informatik, speziell das Teilgebiet des Software-Engineerings (Softwaretechnik), forscht schon länger nach Möglichkeiten, die Lösungen im Entwurfsprozess handhabbar zu machen. Ein Lösungsansatz der Softwaretechnik sind hierfür die *Entwurfsmuster*, die Lösungsvorschriften beziehungsweise Musterlösungen für typische Datenverarbeitungsaufgaben technologieneutral beschreiben und somit eine Wiederverwendung von bewährten Lösungen ermöglichen. Ein Defizit der Entwurfsmuster der Informatik ist die fehlende Einbindung von Anforderungen, die an Systeme gestellt werden.

Das Konzept der Entwurfsmuster stammt ursprünglich aus dem Bereich der Architektur und wurde in Zusammenhang mit der Entwicklung des objektorientierten Entwurfsparadigmas auf die Softwaretechnik übertragen [Gamma et al. 2004]. Darauf aufbauend wurden zahlreiche Forschungsarbeiten zur Konzeption sowie zum Einsatz von Entwurfsmustern veröffentlicht und auf andere Wissenschaftsgebiete, wie beispielsweise das Requirements-Engineering, übertragen.

Auch im Bereich der Automatisierungstechnik hat das Konzept der Entwurfsmuster erste Einsatzmöglichkeiten erhalten [Alznauer et al. 2003]. Charakteristisch hierbei ist, dass

- kein expliziter Bezug zwischen den vorgeschlagenen Lösungen und den Anforderungen hergestellt wird.
- die Lösungen oftmals implementierungsbezogen sind und diese aufgrund der Darstellung der speziellen Lösungen und nicht des Lösungsprinzips keinen Technologiewechsel überdauern.
- Aspekte der Verteilung der Funktionalität in verteilten Automatisierungssystemen nicht oder nur unzureichend behandelt werden.

1.2 Herausforderungen des Engineerings

Herkömmliche Entwurfsmethoden für automatisierte Systeme fokussieren auf zentralistische Strukturen [vgl. EIT Bericht 2010, S. 5 f.]. Auch wenn es inzwischen erste Ansätze für verteilte Automatisierungssysteme gibt, so fehlt es an Beschreibungsmitteln, Methoden und Werkzeugen [Schnieder 1999] [Löwen et al. 2005] für den systematischen Entwurf verteilter Automatisierungssysteme [vgl. EIT Bericht 2010, S. 5 f.]. »Systematisch« bedeutet in diesem Kontext, dass zum einen die funktionalen und nicht-funktionalen Anforderungen [vgl. Harbach et al. 2007, S. 262 f.], die sich aus der Verteilung der Funktionalität ergeben, berücksichtigt werden müssen sowie zum anderen die Wiederverwendung von Lösungen gefördert werden muss [vgl. Fay et al. 2009, S. 83 f.] [vgl. EIT Bericht 2010, S. 5 f.].

Um die Komplexität verteilter Automatisierungssysteme beherrschbar zu machen, müssen die Herausforderungen während des Engineerings herausgestellt werden. In verteilten Automatisierungssystemen steht der Anwendungsentwickler vor der Herausforderung,

- dass eine Verteilung von Automatisierungsfunktionen auf unterschiedliche Komponenten notwendig ist, um die geforderten funktionalen Anforderungen unter Berücksichtigung der nicht-funktionalen Anforderungen zu erfüllen,
- für die projektspezifischen funktionalen Anforderungen eine passende auf mehrere Steuerungen verteilbare Funktionsarchitektur zu finden und
- die Komplexität der Zusammenhänge von verteilten Funktionen sowie deren ausführender Hardware während des Entwurfs zu beherrschen.

Um die Herausforderungen und die Komplexität verteilter Automatisierungssysteme beherrschbar zu machen, müssen die herkömmlichen Entwurfsmethoden des Engineerings optimiert werden.

1.3 Zielsetzung und Aufbau der Arbeit

In der vorliegenden Arbeit widmet sich deren Verfasserin der Entwicklung eines Vorgehensmodells für verteilte Automatisierungssysteme und einer Methode zur Wiederverwendung von bewährten Lösungen. Ziel dieser Arbeit ist es, die Voraussetzungen zu schaffen, um den in

Abschnitt 1.2 beschriebenen Herausforderungen zu begegnen. Hierzu soll der Anwendungsentwickler gezielt beim Entwurf von verteilten Automatisierungssystemen unterstützt werden. Für einen systematischen und anforderungsgetriebenen Entwurf beschreibt die vorliegende Arbeit ein Vier-Ebenen-Vorgehensmodell, das den Entwurf einer verteilten Steuerungsarchitektur unter Berücksichtigung von nfAs unterstützt. Auf Basis dieses Vorgehensmodells wurde ein Musterkonzept entwickelt, das die Wiederverwendung von bewährten Lösungen ermöglicht und den Anwendungsentwickler bei der Verteilung der Funktionalität unterstützt. Dieser Ansatz führt den Anwendungsentwickler mithilfe einer systematischen Vorgehensweise, hilft die Auswirkungen der Designentscheidungen abzuschätzen (durch Merkmale und Entwurfsmuster) und unterstützt den Anwendungsentwickler bei der Entscheidungsfindung. So kann die Erfüllung von nfAs schon während des Entwurfs überprüft werden.

Im Anschluss an diese Einleitung erfolgt in *Kapitel 2* zunächst eine Analyse der Rahmenbedingungen des Engineerings von Automatisierungssystemen. Ziel dieses Kapitels ist die Beschreibung der domänenunabhängigen Grundlagen des Engineerings. Neben einer Begriffsklärung erfolgt eine Beschreibung der wesentlichen Aspekte des Engineerings sowie eine Abgrenzung der unterschiedlichen Steuerungsarchitekturen. Die Analyse dieser Grundlagen erlaubt eine abschließende Betrachtung, die sich im Wesentlichen auf die Steuerungsarchitekturen bezieht.

Ausgehend von Kapitel 2 werden in *Kapitel 3* die Grundlagen des aktuellen Stands der Wissenschaft und Technik im Bereich verteilter Automatisierungssysteme betrachtet. Hierzu werden die wesentlichen Anforderungen erläutert und die verschiedenen Ansätze verteilter Automatisierungssysteme beschrieben. Neben einer Beschreibung der Vorgehensmodelle und Strukturen verteilter Automatisierungssysteme werden die existierenden Lösungsansätze für das Verteilungsproblem beschrieben. Die Analyse existierender Forschungsansätze erlaubt abschließend eine Zusammenfassung und Bewertung des Stands der Wissenschaft und Technik.

In *Kapitel 4* wird der aktuelle Stand der Wissenschaft und Technik im Bereich der Wiederverwendung beschrieben. Nach einer abschließenden Bewertung der Wiederverwendungskonzepte erfolgt eine nähere Betrachtung des Aspekts Entwurfsmuster. Die Analyse existierender Forschungsansätze im Bereich Entwurfsmuster endet mit einer Zusammenfassung und Bewertung.

Zusätzlich zu den vorherigen Kapiteln runden die in *Kapitel 5* diskutierten Forschungsansätze den aktuellen Stand der Wissenschaft und Technik ab und erlauben eine Identifikation und Darstellung des Handlungsbedarfs. Im abschließenden Teil dieses Kapitels werden die wesentlichen Ergebnisse und Erkenntnisse des aktuellen Stands der Wissenschaft und Forschung herausgestellt, mit Fokus auf systematischen anforderungsgetriebenen Entwurf verteilter Automatisierungssysteme sowie einer Wiederverwendung von bewährten Lösungen.

Auf Basis der Ergebnisse der vorherigen Kapitel erfolgt in *Kapitel 6* die Beschreibung der methodischen Grundlagen sowie der Umsetzung des entwickelten Konzepts hinsichtlich einer systematischen Vorgehensweise für den Entwurf verteilter Automatisierungssysteme. Des Weiteren werden die für verteilte Automatisierungssysteme besonders relevanten nfAs beschrieben und in das Vorgehensmodell eingebunden.

Auf Grundlage des in Kapitel 6 vorgestellten Konzepts eines Vorgehensmodells erfolgen in *Kapitel 7* die Beschreibung der methodischen Grundlagen sowie der Umsetzung des entwickelten Konzepts hinsichtlich einer Entwurfsunterstützung in Form von Wiederverwendung. Dieses Konzept der Wiederverwendung wird in das Konzept des Vorgehensmodells integriert und erlaubt dadurch eine systematische Entwurfsunterstützung.

Daran anknüpfend wird in *Kapitel 8* eine beispielhafte Darstellung des Entwurfsmusterkonzepts beschrieben. Auf Basis dieser Darstellung wird ein UML-Modell für die Entwurfsmuster dargestellt, das unterschiedliche Sichten auf die Muster ermöglicht.

In *Kapitel 9* wird anhand eines fertigungstechnischen und verfahrenstechnischen Anwendungsbeispiels aufgezeigt, wie ein systematischer Entwurf verteilter Automatisierungssysteme unter Anwendung der vorgeschlagenen methodischen Vorgehensweise und Entwurfsunterstützung möglich ist. Hierzu werden die einzelnen Arbeitsschritte des Vorgehensmodells und die Anwendung der Entwurfsmuster anhand zweier industrieller Anlagenbeispiele aufgezeigt.

Die wesentlichen Ergebnisse und Erkenntnisse dieser Arbeit werden in *Kapitel 10* zusammengefasst. Ein Ausblick auf weitere zukünftige Entwicklungen und Forschungsfragen im Kontext des vorgestellten Konzepts runden die Arbeit ab.

2 Engineering von Automatisierungssystemen

Das nachfolgende Kapitel beschreibt die Grundlagen des Engineerings von Automatisierungssystemen. Hierzu werden zum einen der Begriff *Engineering* eingeführt und zum anderen die verschiedenen Aspekte des Engineerings. Die Aspekte des Engineerings umfassen die unterschiedlichen Phasen des Engineerings, die Beteiligten sowie Vorgehensmodelle, anhand derer die Beteiligten die Steuerungsarchitekturen entwickeln. Anschließend erfolgt eine Vorstellung der verschiedenen Steuerungsarchitekturen und abschließend eine Schlussfolgerung zum Stand der Wissenschaft und Technik.

2.1 Begriffsklärung

Der Begriff *Engineering* beschreibt die technische Umsetzung einer Anlage und umfasst die Tätigkeiten der Ingenieure von der Planung und Realisierung einer Anlage bis hin zu deren Inbetriebnahme, wobei das Konzept der zu entwickelnden Anlage in Zusammenarbeit mit dem Betreiber erarbeitet wird [vgl. Wagner 2008, S. 68] [vgl. Fay 2009, S. 44]. Darüber hinaus haben die Ingenieure die Aufgabe, aus den zur Verfügung stehenden technischen Möglichkeiten ein System zu erstellen, das die Anforderungen des Betreibers erfüllt, und die Überwachung der Umsetzung zu betreuen [vgl. Wagner 2008, S. 68]. Deshalb umfasst der Begriff *Engineering* zusätzlich auch Tätigkeiten, die während des Betriebs auftreten, wie beispielsweise Überprüfung, Optimierung und Erweiterung [vgl. Fay 2009, S. 44]. In diesem Zusammenhang müssen nicht nur die technischen Aspekte beachtet werden, sondern auch die wirtschaftlichen. Dies stellt oftmals eine große Herausforderung dar, da Komplexität und Größe der zu planenden Anlage stark variieren können. Zum einen können bestehende Anlagen erweitert beziehungsweise geändert werden und zum anderen kann der Ingenieur mit einer vollständigen Planung eines neuen Systems beauftragt werden.

Abhängig von der zu entwickelnden Anlage beziehungsweise vom zu realisierenden technischen Prozess sind für das Engineering unterschiedliche Gewerke sowie Werkzeuge erforderlich. Wesentliche Grundlage für eine erfolgreiche Einbindung der unterschiedlichen Gewerke ist deren Zusammenwirken und die Kommunikation. Dies ist Voraussetzung dafür, dass Fehler frühzeitig erkannt und behoben werden können sowie die Anlage termingerecht fertiggestellt werden kann [vgl. VDI 3695-1 2010, S. 3][#].

2.2 Aspekte des Engineerings der Automatisierungstechnik

In diesem Abschnitt werden zunächst die Phasen des Engineerings sowie die am Engineering-Prozess eingebundenen Beteiligten dargestellt. Darüber hinaus werden unterschiedliche Vorgehensmodelle der Softwaretechnik und der Automatisierungstechnik vorgestellt, die den Engineering-Prozess unterstützen.

2.2.1 Phasen des Engineerings der Automatisierungstechnik in Zusammenhang mit anderen Gewerken

In den Anlagenbau ist eine Gruppe von Personen involviert, die über Wissen aus einem bestimmten Bereich beziehungsweise einem Gewerk oder mehrerer Gewerke verfügt und interdisziplinär zusammenarbeitet. Das Wissen aus den einzelnen Gewerke wird eingesetzt, um die unterschiedlichen Aufgaben zu lösen, die während des Anlagenbaus entstehen. Anzahl und Art der Gewerke sind abhängig von der zu planenden Automatisierungsanlage. Im Engineering von Anlagen wird zwischen Anlagen der Fertigungstechnik (fertigungstechnische Anlage) und der Verfahrenstechnik (prozesstechnische Anlage) unterschieden.

Obwohl der Bau einer Anlage, wie jedes andere Projekt auch, ein individuelles Vorgehen ist, lässt sich die Projektabwicklung systematisch in Phasen unterteilen. Tauchnitz beschreibt in [Tauchnitz 2013], dass das Engineering die gesamte Planungsphase des Lebenszyklus einer Anlage abdeckt. Laut Tauchnitz umfasst der Begriff *Engineering* alle Phasen des gesamten Lebenszyklus einer Anlage, von der Planung über die Betriebsphase bis hin zur Demontage [vgl. Tauchnitz 2013, S. 47]. Marquardt definiert für die Planung prozesstechnischer Anlagen die Phasen Konzeptplanung, Basisplanung, Ausführungsplanung, Errichtung, Inbetriebnahme und Instandhaltung, wobei nur die ersten drei Phasen dem Anlagenengineering zuzuordnen sind [vgl. Marquardt & Nagel 2003, S. 52]. Nach [Schmidtberger 2008] besteht das Engineering einer prozesstechnischen Anlage aus den Phasen Vorplanung, Auslegungsplanung, Entwurfsplanung und Ausführungsplanung [vgl. Schmidtberger 2008, S. 4 ff.].

Bei der Planung prozesstechnischer Anlagen sind die Gewerke Chemie, Verfahrenstechnik, Bautechnik, Maschinentechnik, Rohrleitungsbau, Elektrotechnik, Automatisierungstechnik, Leittechnik und Informationstechnik beteiligt [vgl. Schmidtberger 2008, S. 8]. Drath definiert in [Weidemann & Drath 2010] den grundlegenden Planungsprozess einer prozesstechnischen Anlage nicht gänzlich anders, verwendet aber andere Begrifflichkeiten und Herangehensweisen. Demnach startet der grundlegende Planungsprozess einer prozesstechnischen Anlage mit einer Angebotsphase [vgl. Weidemann & Drath 2010, S. 19 ff.]. Im Anschluss erfolgt die Verfahrensplanung, die von den Mitarbeitern im Gewerk Verfahrenstechnik durchgeführt wird. Außerdem werden die Anforderungen an den Anlagenbau spezifiziert [vgl. Weidemann & Drath 2010, S. 19 ff.]. Das Ergebnis dieser Phase ist beispielsweise das R&I-Fließbild, das der nachfolgenden Phase als Eingangsdokument dient. In dieser Phase erstellen die Mitarbeiter im Gewerk Automatisierungstechnik die Planung der Leittechnik sowie der Automatisierungseinrichtung [Weidemann & Drath 2010, S. 19 ff.]. Nach erfolgreicher Planung von Automatisierungseinrichtung und Leittechnik erfolgt die Installation und Inbetriebnahme [Weidemann & Drath 2010, S. 19 ff.].

In der Fertigungsindustrie beginnt die Planung einer Automatisierungsanlage typischerweise mit einem Produktentwicklungsprozess, und darauf folgt die Fabrikplanung. Mandel definiert für die Planung einer fertigungstechnischen Anlage die Phasen Zielplanung, Vorarbeiten, Grobplanung, Feinplanung sowie Umsetzungsplanung und Umsetzung [vgl. Mandel 2009, S. 12 ff.]. Drath verwendet in [Weidemann & Drath 2010] für die Planung einer fertigungstechnischen Anlage eine Vorgehensweise, die den Planungsprozess in drei Phasen unterteilt [vgl. Weidemann & Drath 2010, S. 16 ff.]. In Phase 1 erfolgen die Produktentwicklung und die Anlagenvorplanung, die unter anderem von den Mitarbeitern der Gewerke Bautechnik und Mechanik durchgeführt werden [vgl. Weidemann & Drath 2010, S. 16 ff.]. Anschließend wird in Phase 2 die Auftragsvergabe umgesetzt, und in Phase 3 erfolgt die Realisierung und Produktion,

die unter anderem in den Gewerken Mechanik, Elektrotechnik, Leittechnik, Informationstechnik und Automatisierungstechnik umgesetzt wird [vgl. Weidemann & Drath 2010, S. 16 ff.]. In der VDI-Richtlinie 3695-1 [VDI 3695-1 2010]# ist der Engineering-Prozess allgemeingültiger definiert. Er wird untergliedert in die Phasen Akquisition, Planung, Realisierung und Inbetriebnahme [vgl. VDI 3695-1 2010, S. 3]#.

Wie zuvor beschrieben, werden die Phasen des Engineering unterschiedlich betrachtet beziehungsweise zusammengefasst. Das Phasenmodell von [Tauchnitz 2013] deckt im weiteren Sinne alle zuvor beschriebenen Möglichkeiten ab und wird im Nachfolgenden verwendet beziehungsweise betrachtet.

2.2.2 Beteiligte des Engineeringprozesses

Grundsätzlich wird bei der Planung einer Anlage unterschieden zwischen dem Anlagenbetreiber, der die zu errichtende Anlage beschafft und betreibt, und dem Anlagenbauer, der nach vereinbartem Liefer- und Leistungsumfang die Planung, Lieferung, Montage und Inbetriebnahme übernimmt [vgl. Helmus 2003, S. 3]. Hierbei müssen Großunternehmen abgegrenzt werden, die oftmals für die Anlagenplanung eine eigene Abteilung haben. In der VDI-Richtlinie 3695-1 sind alle Beteiligten, die beim Engineering einer Automatisierungsanlage mitwirken, unter dem Begriff Engineering-Organisation zusammengefasst. Diese sind zum Beispiel Ingenieurbüros, Zulieferer, Anlagenbauer sowie Anlagenbetreiber [vgl. VDI 3695-1 2010, S. 3]#.

2.2.3 Vorgehensmodelle des Engineerings der Automatisierungstechnik

Um die komplexe Aufgabe des Engineerings erfolgreich durchzuführen, setzt der Bau einer Automatisierungsanlage ein individuelles Vorgehen voraus, da jede Anlage andere Aufgaben, Beteiligte und Rahmenbedingungen besitzt. Trotzdem kann das Engineering systematisch und standardisiert dargestellt und durchgeführt werden. Deshalb wurden Vorgehensmodelle entwickelt, die ein Hilfsmittel darstellen und den Entwickler bei der Planung und Umsetzung einer Automatisierungsanlage unterstützen. Vorgehensmodelle werden somit für eine zielorientierte Festlegung der zukünftigen Schritte angewandt und legen mögliche Ergebnisse der einzelnen Schritte fest, was zu effektiven und effizienten Ergebnissen führt [vgl. Lindemann 2009, S. 33]. Für den systematischen Entwurf sowohl im Maschinenbau als auch in der Softwaretechnik existieren eine Vielzahl von Vorgehensmodellen. Diese werden in der Regel in domänenübergreifende und domänenspezifische Vorgehensmodelle unterteilt. Die domänenübergreifenden Vorgehensmodelle sind in ihrer Gültigkeit nicht auf bestimmte Anwendungsgebiete beschränkt, im Gegensatz zu den domänenspezifischen Vorgehensmodellen [vgl. Vogel-Heuser 2003, S. 34 f.]. Die meisten domänenübergreifenden Vorgehensmodelle, wie beispielsweise das V-Modell oder das V-Modell XT®, kommen aus der Entwicklung mechatronischer Systeme und der Softwaretechnik. Dies resultiert daraus, dass gerade in der Mechatronik und der Softwaretechnik fachgebietübergreifende Zusammenarbeit und Kommunikation notwendig sind. In den domänenspezifischen Anwendungsgebieten wurden Konzepte und Vorgehensweisen entwickelt, die auf ihr Fachgebiet spezialisiert sind, wie beispielsweise das NAMUR-Vorgehensmodell.

Anhand von Kriterien erfolgt eine abschließende Bewertung (siehe Tabelle 2.1 auf Seite 14) der nachfolgend beschriebenen Vorgehensmodelle, um aufzuzeigen, dass existierende Vorgehensmodelle für das effiziente Engineering von verteilten Automatisierungssystemen nicht ausreichend sind.

NA 35

Das von der NAMUR erarbeitete Arbeitsblatt 35 [NA 35 2003]# (Standardstrukturplan siehe Anhang A, Tabelle A.1 auf Seite 126) ist keine Norm oder Richtlinie, sondern eine Handlungsempfehlung für die Abwicklung und Projektierung von leittechnischen Projekten auf Basis von Industrieerfahrung und Forschungsprojekten. Das domänenspezifische NA 35 beschreibt die Projektierung von Prozessleitsystemen (PLS), wobei die Phasen mit der Projektierung der gesamten Anlage vergleichbar sind. Die NAMUR empfiehlt die Anwendung der Handlungsempfehlung für PLS in der chemischen Verfahrenstechnik. Sie kann aber auch für andere PLS verwendet werden.

Die NA 35 beinhaltet einen Standard-Projektstrukturplan mit den drei Grundpfeilern Projektierung, Qualitätsmanagement sowie Projektmanagement, die parallel ablaufen. Dieser Standard-Projektstrukturplan beschreibt eine detaillierte Vorgehensweise für das Engineering einer Anlage und ist im Grundpfeiler Projektierung in sieben Phasen unterteilt. Diese sieben Phasen reichen von der Grundlagenermittlung bis hin zum Projektabschluss und sind in weitere Tätigkeiten untergliedert. Für die Umsetzung dieser Tätigkeiten beinhaltet das NA 35 einen Methodenkatalog, der für jede Tätigkeit die benötigten Inputs, Umsetzungsschritte sowie Outputs spezifiziert. Der Input umfasst alle benötigten Dokumente und Informationen, um die Umsetzungsschritte durchführen zu können. In der Umsetzung sind die relevanten Einzelaktivitäten aufgelistet, die in den entsprechenden Tätigkeiten durchgeführt werden müssen. Die einzelnen Tätigkeiten einer Phase können parallel abgearbeitet werden. Des Weiteren spezifiziert die Umsetzung Beteiligte, Methoden, Hilfsmittel und Projektierungsaufwand. Der Output beinhaltet alle Ergebnisse dieser Tätigkeit, die wiederum als Input für andere Unterphasen dienen können. Sobald die Umsetzungsschritte durchgeführt und somit die Ergebnisse einer Phase erreicht wurden, kann in eine nachfolgende Phase übergegangen werden. Die Grundpfeiler Qualitätsmanagement und Projektmanagement sind Bestandteil des gesamten Planungsprozesses. Sie laufen während dieser Zeit begleitend zur Projektierung.

Der Standard-Projektstrukturplan der Handlungsempfehlung NA 35 ermöglicht somit eine Untergliederung der Arbeitsabläufe in einzelne Phasen sowie Tätigkeiten und macht dadurch eine strukturierte Herangehensweise an die Planung der PLT einer Anlage möglich. Durch das begleitend einsetzende Qualitäts- und Projektmanagement können Fehler frühzeitig erkannt und beseitigt werden. Die NA 35 beschreibt sehr detailliert, wie bei der Planung der PLT einer Anlage vorzugehen ist. Die jeweiligen Einzelaktivitäten sind konkret festgelegt und erlauben nur einen geringen Freiheitsgrad. Im Vorgehensmodell selbst findet sich keine Aussage, ob dessen Inhalte angepasst werden können. Die Schnittstellen der Tätigkeiten nach außen sind jeweils durch Eingangs- und Ausgangsdokumente beschrieben. Die NA 35 sieht keine Iterationen (Rücksprünge) zwischen den Phasen vor und erlaubt eine parallele Durchführung der Tätigkeiten in den Phasen, wobei die einzelnen Phasen sequenziell durchgeführt werden.

Entwicklungsmethodik nach VDI 2206

Die VDI 2206 [VDI 2206 2004]# ist eine Richtlinie für eine domänenspezifische flexible Vorgehensweise zur Entwicklung mechatronischer Systeme. Der Grundgedanke der Richtlinie ist, dass es keine optimale Form für eine Entwicklungsmethodik eines Konstruktionsprozesses gibt und an jedes mechatronische System andere Anforderungen gestellt werden müssen. Die Richtlinie besteht im Wesentlichen aus den Elementen Problemlösungszyklus, V-Modell und vordefinierte Prozessbausteine.

Der Problemlösungszyklus beschreibt eine zyklisch ablaufende Handlungsanweisung, um den Anwendungsentwickler bei der Lösung von Problemstellungen zu unterstützen. Der Problemlösungszyklus wird in die Vorgehensweisen *Ist-Zustand orientiertes Vorgehen* sowie *Soll-Zustand orientiertes Vorgehen* unterteilt. In der *Ist-Zustand orientierten Vorgehensweise* werden zuerst die vorhandenen Strukturen und Möglichkeiten analysiert, um daraufhin das eigene Ziel zu formulieren. In der *Soll-Zustand orientierten Vorgehensweise* wird ein vorgegebenes Ziel übernommen und analysiert, welche Ausgangssituation vorliegt, um das Problem zu lösen. Aufgrund dieser Vorgehensweisen kann die VDI 2206 für eine produktorientierte und eine problemorientierte Entwicklung verwendet werden. Die Abläufe werden im Schritt Synthese und Analyse zusammengeführt, in welchem Lösungskonzepte erstellt und verbessert werden. Im Anschluss daran werden die Lösungskonzepte analysiert und bewertet, um im nachfolgenden Schritt eine Entscheidung zu treffen. Der Zyklus muss erneut beginnen, wenn die gestellten Anforderungen nicht erfüllt werden können. Diese problemorientierte Vorgehensweise wird auf der Mikroebene – bei kurzfristigen Projektverläufen – angewendet.

Das zweite Element der VDI 2206 ist das V-Modell (siehe Anhang A, Abbildung A.1 auf Seite 127), das in Phasen unterteilt ist. Diese Phasen werden in der Richtlinie Prozessbausteine genannt und stellen somit das dritte Element der Entwicklungsmethodik dar. Die Entwicklungsmethodik nach VDI 2206 beginnt mit der Definition der Anforderungen, die den Ausgangspunkt für das Vorgehensmodell bilden. Anhand der Anforderungen wird das spätere Produkt bewertet. Die erste Phase beziehungsweise den ersten Prozessbaustein des Vorgehensmodells bildet der Systementwurf. Mit ihm verfolgt der Einwickler das Ziel, ein domänenübergreifendes Lösungskonzept festzulegen. In der Phase des domänenspezifischen Entwurfs wird das Lösungskonzept detailliert. Dies erfolgt zumeist getrennt in den unterschiedlichen Domänen. Die Phase der Systemintegration fasst alle Ergebnisse der einzelnen Domänen zu einem Gesamtsystem zusammen. Dadurch soll das Zusammenwirken der einzelnen Domänen untersucht werden. In der Phase der Eigenschaftenabsicherung wird der Entwurfsfortschritt anhand des Lösungskonzepts sowie der Anforderungen überprüft. Modellbildung und Modellanalyse sind Phasen, die alle Phasen des Vorgehensmodells begleiten. Das Produkt ist das Ergebnis des Vorgehensmodells. Durch sukzessives Durchlaufen des Vorgehensmodells kann das mechatronische System zunehmend konkretisiert werden. Diese Vorgehensweise, basierend auf einem V-Modell, wird auf der Makroebene – bei mittel- und langfristigen Projektverläufen – angewendet.

Die VDI 2206 erlaubt eine begrenzte Anpassbarkeit des Vorgehensmodells, weil die einzelnen Aktivitäten in den Phasen nicht an die Problemstellungen und Herausforderungen verteilter Automatisierungssysteme angepasst werden können. Der Ablauf der jeweiligen Phasen sowie deren Aktivitäten bieten keine Möglichkeit zu inhaltlichen Anpassungen. Jedoch ist mit der Phase des domänenspezifischen Entwurfs ein Vorgehensschritt definiert, in dem ein etabliertes domänenspezifisches Vorgehensmodell genutzt werden kann. Mit diesem Freiheitsgrad kann der Anwendungsentwickler automatisch dessen Inhalt bestimmen und anpassen. Die Entwicklungsmethodik definiert Phasen und Aktivitäten, die allerdings lediglich im domänenspezifischen Entwurf einen gewissen Freiheitsgrad und eine individuelle Vorgehensweise ermöglichen. Die VDI 2206 definiert keine Schnittstellen zwischen den einzelnen Phasen. Iterationen sind lediglich in der späteren Systemintegration erlaubt und ermöglichen dann einen Rücksprung zum Systementwurf. Tätigkeiten sowie Phasen im domänenspezifischen Entwurf können parallel durchlaufen werden. Alle weiteren Phasen und Tätigkeiten werden sequenziell ausgeführt.

V-Modell

Das V-Modell (siehe Anhang A, Abbildung A.2 auf Seite 127) ist ein international anerkanntes domänenübergreifendes Vorgehensmodell für die Projektabwicklung [vgl. Bender 2005, S. 31]. Es entstand im Hinblick auf die Softwaretechnik und ist eine weit verbreitete Vorgehensweise für die Software-Entwicklung.

Ein wesentlicher Fortschritt des V-Modells im Vergleich zu klassischen Vorgehensmodellen, wie zum Beispiel dem Wasserfallmodell, ist die Einbindung der Qualitätssicherung in den Entwicklungsprozess. Hierzu findet durch die V-förmige Anordnung eine Überprüfung der Ergebnisse des rechten Astes anhand der Spezifikationen des linken Astes statt [vgl. Vogel-Heuser 2003, S. 26]. Die jeweiligen Phasen des V-Modells können in Aktivitäten unterteilt werden, die sich nicht nur auf die Softwareerstellung, sondern auch auf die benötigte Hardware beziehen [vgl. Bender 2005, S. 32]. Das Ziel des V-Modells ist eine möglichst frühe Definition von Anwendungs- sowie Testfällen (linker Ast), die in späteren Phasen (rechter Ast) durch Modul-, Integrations-, System- und Abnahmetests verifiziert und validiert werden [vgl. Bender 2005, S. 32].

Das V-Modell ermöglicht eine individuelle Vorgehensweise und eine Anpassung, sodass einzelne Aktivitäten in den Phasen definiert werden können, die sich speziell an den Problemstellungen verteilter Automatisierungssysteme orientieren. In keiner Phase werden explizite Aktivitäten definiert. Dies führt dazu, dass bei der Anwendung des V-Modells sehr flexibel entschieden werden kann, wie ein Projekt umgesetzt wird und welche Methoden genutzt werden. Die Nachteile sind jedoch der hohe Anpassungsaufwand und dass das Vorgehensmodell keine Hilfestellung bei der Umsetzung gibt, beispielsweise welche Aktivitäten in bestimmten Projekten relevant sind. Das V-Modell ist nicht nur für die Anwendung von Softwareprojekten, sondern auch für die gesamte Systementwicklung und auch im Engineering von Automatisierungsanlagen anwendbar [vgl. Vogel-Heuser 2003, S. 28].

Das V-Modell definiert keine Schnittstellen zwischen den einzelnen Phasen, aber diese bauen aufeinander auf. Die einzelnen Phasen werden sequenziell bearbeitet. Iterationen sind lediglich vom rechten Ast zum linken erlaubt. Eine parallele Durchführung von Tätigkeiten in den Phasen wird dadurch ermöglicht, dass keine konkreten Tätigkeiten definiert werden.

V-Modell xt®

Das V-Modell extreme tailoring (V-Modell xt) [V-Modell xta][@] (siehe Anhang A, Abbildung A.3 auf Seite 128) ist eine Weiterentwicklung des V-Modells und eine domänenübergreifende Vorgehensweise für die Planung und Durchführung von Projekten [vgl. Rausch & Broy 2008, S. 2 ff.]. Das V-Modell xt wird als ein Vorgehensmodell verstanden, das für viele verschiedene Projektkonstellationen anwendbar ist und an die Projektbedingungen angepasst werden kann [vgl. Rausch & Broy 2008, S. 6]. Diese Anpassung auf die projektspezifischen Bedürfnisse wird Tailoring genannt.

Das V-Modell xt definiert detailliert die Vorgehensweise sowie die jeweils zu erstellenden Ergebnisse. Des Weiteren werden die jeweiligen Verantwortlichen festgelegt und spezifiziert »wer«, »wann«, »was« zu erledigen hat [vgl. V-Modell xta, S. 6][@]. Hierzu wird das V-Modell in drei Projekttypen klassifiziert, die für die verschiedenen Projektkonstellationen ausgelegt sind. Die Vorgehensweise in den Projekttypen wird beschrieben. Diese sind ein wesentlicher Bestandteil des V-Modell xt und der erste Schritt zur Festlegung der Aufgaben. Jeder Projekttyp

besteht wiederum aus mindestens einer Projekttypvariante, welche die Rahmenbedingungen für den möglichen Ablauf des Projekts näher beschreibt. Zusätzlich können sogenannte Vorgehensbausteine in das Projekt integriert werden. Diese Vorgehensbausteine beschreiben Produkte, Aktivitäten und Rollen, die für die Aufgabenstellung relevant sind.

Das Produkt im Sinne des V-Modell xt ist nicht nur das Endprodukt, sondern auch das Zwischenprodukt beziehungsweise Zwischenergebnis, und es ist das Ergebnis einer Aktivität. Die Reihenfolge der Aktivitäten in den jeweiligen Vorgehensbausteinen kann individuell angepasst werden. Um dennoch einen geordneten Projektablauf sicherstellen zu können, stehen sogenannte Projekt-Durchführungsstrategien zur Verfügung. Die Projekt-Durchführungsstrategie legt die Reihenfolge der durchzuführenden Aktivitäten und der zu erstellenden Produkte fest. Ein weiterer wichtiger Aspekt des Vorgehensmodells ist der Entscheidungspunkt, der nach Abschluss eines Projektabschnitts gesetzt wird. Er ist vergleichbar mit einem Meilenstein, an welchem das Projekt evaluiert wird. Eine nähere Beschreibung des V-Modell xt ist [Friedrich et al. 2009] zu entnehmen.

Das V-Modell xt ermöglicht somit eine individuelle projektbezogene Vorgehensweise. Aufgrund der komplexen Struktur des V-Modells ist eine Anwendung oftmals schwierig und nur erfahrenen Projektleitern möglich. Ein großer Nachteil des V-Modell xt ist das Änderungsmanagement. Ist eine Änderung in den Anforderungen erforderlich, so müssen vorherige Ergebnisse geändert werden, was sehr viel Zeit kostet. Die Nachteile sind somit der Anpassungsaufwand und der Aufwand zur Anwendung beziehungsweise das Verstehen des Vorgehensmodells durch alle Projektbeteiligten.

Das V-Modell xt beschreibt Schnittstellen der Phasen jeweils durch Eingangs- und Ausgangsdokumente. Das Vorgehensmodell sieht keine Iterationen zwischen den Phasen vor und erlaubt keine parallele Durchführung der Tätigkeiten in den Phasen, da die Vorgehensbausteine aufeinander aufbauen. Die einzelnen Phasen werden sequenziell durchlaufen.

3-Ebenen-Vorgehensmodell

Das 3-Ebenen-Vorgehensmodell [Bender 2005] (siehe Anhang A, Abbildung A.4 auf Seite 128) ist ein domänenspezifisches Vorgehensschema zur Entwicklung mechatronischer Produkte und entstand im Hinblick auf Anforderungen an heutige Entwicklungsprozesse. Die Grundlagen für die Entstehung des 3-Ebenen-Vorgehensmodells entstanden im Projekt equal [equal], das Qualitätssicherungsmaßnahmen in der Praxis untersuchte. Das Vorgehensmodell ist in Ebenen unterteilt und gliedert die unterschiedlichen Domänen in seinem Entwicklungsverlauf. Im 3-Ebenen-Vorgehensmodell sollen neue Anforderungen, wie beispielsweise gestiegene Anzahl der beteiligten Domänen und Softwareanteil sowie der Aspekt Qualitätssicherung, an eingebettete Systeme berücksichtigt werden.

Das Vorgehensmodell besteht für die Domänen Software, Hardware und Mechanik aus drei Ebenen (Systemebene, Subsystemebene und Komponentenebene). Die einzelnen Ebenen werden für jede Domäne V-förmig durchlaufen. Anhand der Prüf- und Integrationstätigkeiten des rechten Astes werden die Entwicklungsebenen des linken Astes getestet. Die Grenzen der Ebenen beinhalten Iterationsschritte, die es beispielsweise ermöglichen, wichtige Komponenten vollständig zu konkretisieren und diese Ergebnisse als Ausgangspunkt für die Entwicklung anderer Komponenten zu nutzen. Das Vorgehensmodell startet in der Systemebene mit einer System-Anforderungsanalyse. Hieraus erfolgt der Systementwurf, der die Hauptfunktionen sowie deren Abhängigkeiten darstellt.

Auf Basis des Systementwurfs erfolgt in der Subsystemebene eine Aufteilung des Systems in miteinander verbundene Subsysteme. Die einzelnen Subsysteme werden an die jeweils relevanten Domänen übergeben. Innerhalb des Subsystems findet nochmals eine Analyse der Anforderungen und ein daraus folgender Entwurf des Subsystems statt. Eine Besonderheit des Vorgehensmodells ist, dass die Informationstechnik (IT) erst nach einer nochmaligen Analyse und einem daraus folgenden Entwurf in die Domänen Hardware und Software unterteilt wird. Die Komponenten-Ebene realisiert die eigentliche Umsetzung des Produkts. Hierzu werden alle Subsysteme in einzelne Komponenten unterteilt und innerhalb ihrer Domäne bearbeitet. Nachdem alle Komponenten eingesetzt beziehungsweise implementiert wurden, erfolgt deren Test. Dies bedeutet, dass bereits in den Entwicklungsphasen für das Gesamtsystem, das Subsystem sowie die Komponenten Testfälle definiert und erarbeitet werden. Nach Abschluss der Testphase erfolgt eine Integration der Komponenten zu Subsystemen, wiederum innerhalb der entsprechenden Domäne. Anschließend findet die Integration des IT-Subsystems mit der Mechanik, die wiederum getestet wird, statt. Im Anschluss daran wird auf der System-Ebene eine Integration aller Subsysteme zu einem Gesamtsystem vollzogen. Diese wird abschließend getestet.

Das Vorgehen zur Umsetzung einer Komponente ist beim 3-Ebenen-Vorgehensmodell frei wählbar. Des Weiteren sind in keiner Phase explizite Aktivitäten definiert, sodass der Anwendungsentwickler flexibel entscheiden kann, welche Methoden für die Projektumsetzung verwendet werden sollen. Der Nachteil ist, dass das Vorgehensmodell keine Hilfestellung bei der Umsetzung gibt, beispielsweise welche Aktivitäten in bestimmten Projekten relevant sind. Das Vorgehensmodell besteht aus oberflächlichen Strukturelementen, in denen die Phasen schwach definiert sind und beinhaltet keine beschriebenen Aktivitäten. In der Komponentenebene ist das Vorgehen noch schwächer definiert, da hier etablierte Vorgehensmodelle eingesetzt werden sollen. Das 3-Ebenen-Vorgehensmodell bleibt in seinen Aussagen sehr allgemein und definiert nur grob das Vorgehen der Phasen. Dies reicht zwar, um die Idee dahinter zu vermitteln, bietet dem Anwendungsentwickler aber darüber hinaus wenig Hilfestellung.

Das 3-Ebenen-Vorgehensmodell beschreibt keine Schnittstellen zwischen den Phasen und beinhaltet keine Iterationen (Rücksprünge) zwischen den Ebenen. Das Vorgehensmodell erlaubt eine parallele Durchführung der Phasen, wobei die Tätigkeiten in den Phasen nicht näher beschrieben sind. Dadurch ist eine parallele Bearbeitung der Tätigkeiten möglich.

Bewertung der Vorgehensmodelle anhand von Kriterien

Anhand von Kriterien, die Vorgehensmodelle in der realen Anwendung erfüllen müssen, erfolgt in Tabelle 2.1 auf der nächsten Seite eine abschließende Bewertung der zuvor beschriebenen Vorgehensmodelle. Diese Bewertung zeigt auf, dass existierende Vorgehensmodelle für das effiziente Engineering von verteilten Automatisierungssystemen (siehe Kapitel 5) nicht ausreichend sind. Vorgehensmodelle für verteilte Automatisierungssysteme müssen Schnittstellen, die für den Informations- und Datenaustausch zwischen den Phasen notwendig sind, vorgeben und Iterationen, die einen Rücksprung in vorherige Phasen ermöglichen, bereitstellen. Des Weiteren ist die parallele Durchführung von Tätigkeiten in den jeweiligen Phasen sowie eine parallele Durchführung der einzelnen Phasen notwendig. Die nachfolgende Tabelle 2.1 zeigt die partielle Erfüllung der Kriterien in den einzelnen zuvor beschriebenen Vorgehensmodellen.

Tabelle 2.1: Bewertung der Vorgehensmodelle anhand von Kriterien

	Schnittstellen	Iterationen	Parallele Tätigkeiten	Parallele Phasen
NA 35	●	○	●	○
VDI 2206	○	◐	◐	◐
V-Modell	○	◐	●	○
V-Modell xt	●	○	○	○
3-Ebenen-Vorgehensmodell	○	○	●	●

● Kriterien erfüllt ◐ Kriterien teilweise erfüllt ○ Kriterien nicht erfüllt

2.3 Abgrenzung der Steuerungsarchitekturen

In diesem Abschnitt werden die unterschiedlichen Architekturen eines Automatisierungssystems vorgestellt. Hierzu erfolgt eine Beschreibung der Steuerungsarchitekturen in ihrer zentralen, dezentralen und verteilten Ausrichtung sowie deren Struktur. Die Begrifflichkeiten der dezentralen und verteilten Steuerungsarchitektur werden in der Literatur oftmals synonym verwendet, mit der Folge, dass keine Unterscheidung zwischen diesen zwei Architekturen erfolgt [Papenfort 2006] [Göhner et al. 2003]. Im Gegensatz hierzu unterscheidet die IEC 61 499 [IEC 61 499-1 2006][#] die Begrifflichkeiten der dezentralen und verteilten Steuerungsarchitektur auf Basis der Software beziehungsweise der Entscheidungsfindungsebene. In beiden Steuerungsarchitekturen wird die Software in Anwendungen unterteilt.

Bei einer dezentralen Steuerungsarchitektur ist laut IEC 61 499 eine Anwendung auf einer Hardware-Komponente implementiert, wohingegen bei einer verteilten Steuerungsarchitektur die Anwendung über mehrere Hardware-Komponenten verteilt werden kann. Von Aspern und Diedrich definieren in [von Aspern 2009] und [Diedrich et al. 2011] zusätzlich zur Verteilung der Software auch eine Verteilung der Hardware. Die Definitionen der IEC 61 499 sowie von Aspern und Diedrich werden im Nachfolgenden aufgegriffen und näher erläutert.

2.3.1 Zentrale Steuerungsarchitektur

Eine zentrale Steuerungsarchitektur besteht aus einem abgeschlossenen Anlagenmodul mit nur einer Steuerung [vgl. Diedrich et al. 2011, S. 428]. Diese zentrale Steuerung beinhaltet die gesamte Steuerintelligenz und muss somit eine für die Automatisierungsaufgabe ausreichende Rechenleistung bereitstellen [vgl. Papenfort 2006, S. 43]. In der Regel besteht eine zentrale Steuerungsarchitektur aus einer Steuerung und einer Vielzahl an Feldgeräten, die über eine Parallelverdrahtung oder über Feldbus direkt mit dieser Steuerung verbunden sind [vgl. Diedrich et al. 2011, S. 429] [vgl. von Aspern 2009, S. 42]. Bei einer Parallelverdrahtung bedeutet dies, dass jedes Prozesssignal – Signale der Sensoren und der Aktoren – über eigene Leitungen separat vom Prozess zur zentralen Steuerung und von der zentralen Steuerung zum Prozess geleitet wird [vgl. von Aspern 2009, S. 42]. Diese Möglichkeit der Parallelverdrahtung hat aufgrund des Verdrahtungsaufwands hohe Installationskosten sowie einen hohen Platzbedarf zur Folge [vgl. von Aspern 2009, S. 42]. Die Realisierung einer zentralen Steuerungsarchitektur mittels Feldbus hat den Vorteil, dass der Verdrahtungsaufwand reduziert wird, da die Sensor- und Aktorsignale über einen Feldbus übertragen werden, und die zentrale Steuerung lediglich für jedes Signal eine Variable benötigt beziehungsweise bereitstellen muss. Die Steuerintelligenz

ist hierbei immer noch zentral auf dieser Steuerung realisiert. Eine allgemeingültige zentrale Steuerungsarchitektur – unabhängig von einer Parallelverdrahtung oder Feldbus – ist in Abbildung 2.1 dargestellt.

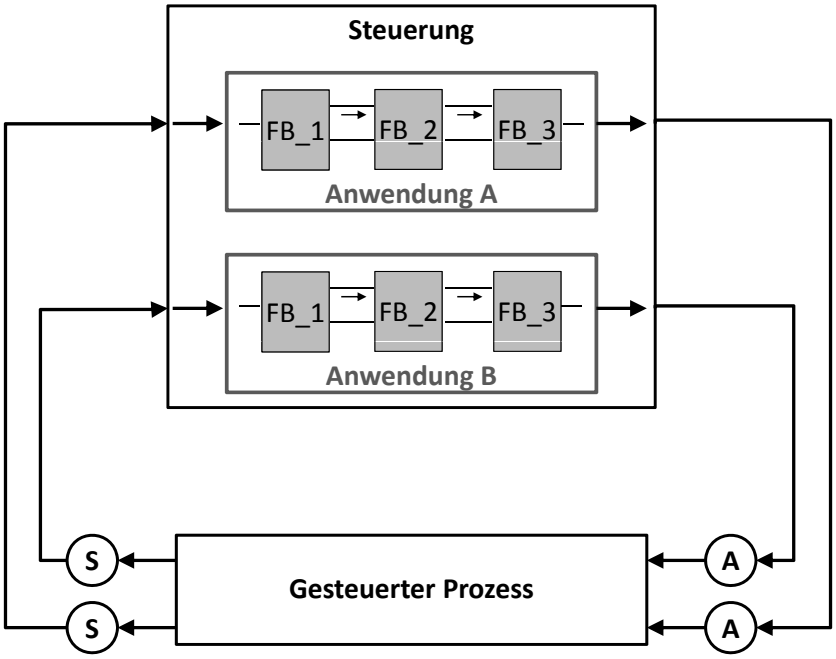

Abbildung 2.1: Zentrale Steuerungsarchitektur

Die komplexe Steuerintelligenz auf einer zentralen Hardware-Komponente hat zur Folge, dass die Gesamtimplementierung eines Systems zwar relativ einfach vorgenommen werden kann, aber aufgrund des hochkomplexen und unübersichtlichen monolithischen Softwareblocks, anstelle mehrerer einfacherer, schlecht wartbar und schlecht wiederverwendbar ist [vgl. Diedrich et al. 2011, S. 428] [vgl. Papenfort 2006, S. 43]. Aufgrund dieser zentralen Implementierung der Steuerintelligenz entstehen große Programme, die eine hohe Prozessorleistung erfordern, um die gewünschten Reaktionszeiten sowie Echtzeitanforderungen zu garantieren [vgl. von Aspern 2009, S. 43]. Wegen der komplexen und fest verbundenen Softwarestrukturen haben zentrale Steuerungssysteme bei Ausfall der zentralen Hardware-Komponente einen Totalausfall des Gesamtsystems zur Folge. Diesen Nachteilen stehen die Vorteile einer relativ schnellen und einfachen Erstinbetriebnahme, eines einfachen Startens und Stoppens des Systems, einer einfachen Archivierung und Ablage sowie der Ausschluss eines Kommunikationsoverheads gegenüber [vgl. Frank et al. 2012a, S. 294 f.]*.

2.3.2 Dezentrale Steuerungsarchitektur

Mit der Entwicklung von Feldbussen, welche die industriellen Anforderungen der Automatisierungstechnik an digitale Kommunikationsverbindungen erfüllen, wurde das Paradigma der dezentralen Steuerungsarchitekturen möglich [vgl. von Aspern 2009, S. 43]. Hierzu wurden Automatisierungssysteme in modulare Strukturen zerlegt und somit dezentrale Steuerungsarchitekturen geschaffen [vgl. Diedrich et al. 2011, S. 429]. Im Hinblick auf die Steuerintelligenz werden große Programme in kleinere funktional abgeschlossene Anwendungen zerlegt und diese jeweils

unterschiedlichen Hardware-Komponenten zugeordnet. Diese räumlich verteilten Hardware-Komponenten beinhalten Anwendungen, die nicht über mehrere Hardware-Komponenten verteilt werden sollen und in ihrer Funktionalität abgeschlossen sind [vgl. Favre-Bulle 2004, S. 87]. Weil die einzelnen Steuerungen abgeschlossene Anwendungen ausführen, ist die Steuerintelligenz und somit die Rechenleistung im Automatisierungssystem verteilt [vgl. Favre-Bulle 2004, S. 250]. Für die Anbindung der Feldgeräte gibt es in einer dezentralen Steuerungsarchitektur eine Vielzahl von Möglichkeiten, wobei zu beachten ist, dass diese Architektur aus mehreren Steuerungen sowie jeweils monolithischen Anwendungen besteht, die zur Erfüllung ihrer jeweiligen Aufgaben nicht miteinander kommunizieren müssen. Diese Möglichkeiten der Anbindung von Feldgeräten sind unter anderem (wie in verteilten Automatisierungssystemen):

1. Anbindung der Sensoren und Aktoren an die Steuerungen mittels einer Parallelverdrahtung
2. Anbindung der Sensoren und Aktoren an Buskoppler mittels einer Parallelverdrahtung; Verbindung dieser Buskoppler mit der Steuerung mithilfe von Feldbus
3. Anbindung der Sensoren und Aktoren mittels Feldbus an die jeweiligen Steuerungen

Zu beachten ist, dass die einzelnen Sensoren und Aktoren über keine Intelligenz zum Ausführen von Steuerungsaufgaben verfügen und lediglich Signale an die Steuerung weiterleiten beziehungsweise Signale von der Steuerung empfangen. Die nachfolgende Abbildung 2.2 zeigt einen allgemeingültigen Aufbau einer dezentralen Steuerungsarchitektur, unabhängig von der Anbindung der Feldgeräte (mit oder ohne Buskoppler).

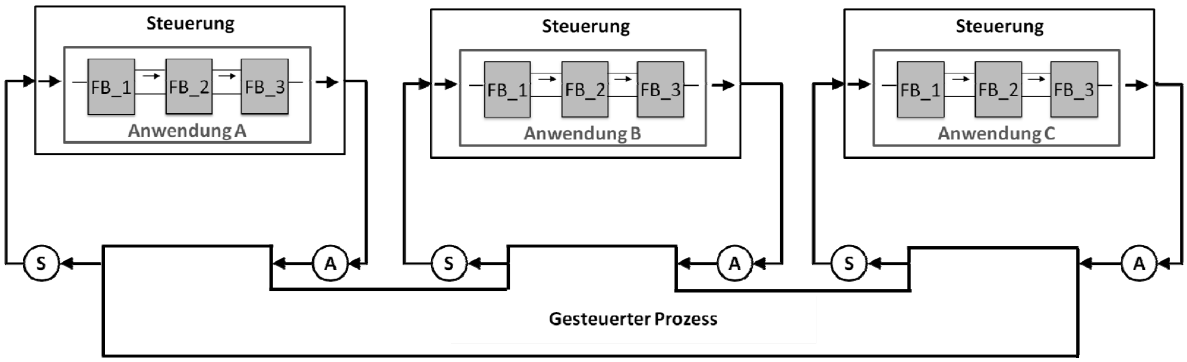

Abbildung 2.2: Dezentrale Steuerungsarchitektur

Im weiteren Verlauf wird die Kombination von Buskoppler und Feldbus in einer dezentralen Steuerungsarchitektur näher erläutert. Diese Systeme sind dadurch gekennzeichnet, dass die Signalein- und Signalauskopplung weg von der Steuerung und somit näher am Prozess angesiedelt wird [vgl. von Aspern 2009, S. 43]. Hierzu müssen laut Aspern die Ein- und Ausgänge direkt mit einem Buskoppler verbunden werden, der einen Datenaustausch über ein Feldbussystem mit ein oder mehreren Steuerungen ermöglicht [vgl. von Aspern 2009, S. 43 f.]. Zwischen dem Buskoppler sowie dem Prozess (Sensoren und Aktoren) erfolgt zum einen, wie in der zentralen Steuerungsarchitektur erläutert, eine Parallelverdrahtung und zum anderen eine serielle Datenübertragung mit nur einer Leitung zwischen dem Buskoppler und der Steuerung mittels Feldbus [vgl. von Aspern 2009, S. 44]. Zwischen der Steuerung und dem Buskoppler entsteht hierbei oftmals eine Master-Slave-Beziehung.

Anhand Abbildung 2.2 auf der vorherigen Seite ist ersichtlich, dass die jeweiligen Anwendungen monolithisch auf den einzelnen Steuerungen implementiert wurden und vor allem zur Erfüllung ihrer Aufgabe keine Kommunikation mit anderen Steuerungen notwendig ist. Diese abgeschlossenen Funktionalitäten auf verschiedenen Hardware-Komponenten ermöglichen Flexibilität bei Änderungen, eine einfachere Störfalllokalisierung sowie die Möglichkeit, Operationen nebenläufig auszuführen. Diesen Vorteilen steht der Nachteil gegenüber, dass ein zusätzlicher Kommunikationsaufwand notwendig ist, wenn die Hardware-Komponenten untereinander kommunizieren müssen. Des Weiteren werden bei dieser Steuerungsarchitektur – im Gegensatz zur zentralen Steuerungsarchitektur – mehrere Hardware-Komponenten benötigt.

2.3.3 Verteilte Steuerungsarchitektur

Aufgrund großer Fortschritte im Bereich der Automatisierungstechnik (technisch weiterentwickelte und leistungsstärkere Geräte sowie neue Methoden und Verfahren) lassen sich neue Systeme und Systemarchitekturen entwickeln. Während in der Vergangenheit hauptsächlich zentrale und dezentrale Architekturen Anwendung fanden, geht der heutige Trend durch die seit 2000 eingeführte verteilte Steuerungsarchitektur (siehe Abbildung 2.3 auf Seite 19) in Richtung Verteilung von Funktionalität [vgl. Favre-Bulle 2004, S. 18]. Es ist anzumerken, dass die Begriffe dezentral und verteilt in der Literatur oftmals bedeutungsgleich verwendet werden. Wie in Abschnitt 2.3.2 erläutert, erfolgt in einer dezentralen Steuerungsarchitektur, wie auch in einer verteilten, eine räumliche Trennung von Steuerungen. Im Gegensatz zu dezentralen Systemen ist in einer verteilten Steuerungsarchitektur die Anwendung auf unterschiedliche Steuerungen verteilt und nicht mehr monolithisch auf einer einzelnen [vgl. Favre-Bulle 2004, S. 87]. Dies hat zur Folge, dass in einer verteilten Steuerungsarchitektur die einzelne Anwendung auf unterschiedliche Hardware-Komponenten verteilt ist, und zur Erfüllung ihrer Aufgabe müssen die Hardware-Komponenten miteinander über ein Kommunikationsnetzwerk kommunizieren [vgl. von Aspern 2009, S. 46]. Hierzu werden die einzelnen Anwendungen auf mehrere Automatisierungsgeräte – Steuerungen und intelligente Feldgeräte – im Feld verteilt und erfüllen einen Teil der Gesamtaufgabe [vgl. von Aspern 2009, S. 46]. Wurmus definiert eine verteilte Steuerungsarchitektur wie folgt: »Unter einem verteilten Automatisierungssystem werden hier mehrere, miteinander vernetzte, programmierbare Komponenten verstanden. Die Komponenten können mit dem zu automatisierenden Prozess verbunden sein, oder auch nicht.« [Wurmus & Wagner 2000, S. 1]

Die IEC 61 499 [IEC 61 499-1 2006]# beschreibt, dass ein verteiltes Automatisierungssystem aus einer Zusammenstellung von Geräten besteht, die miteinander verbunden sind und über ein Kommunikationsnetzwerk kommunizieren. Demnach kann eine Anwendung über ein oder mehrere Geräte verteilt werden, die über Netzwerke miteinander verbunden sind. Eine Kommunikation zwischen den einzelnen Geräten findet dann über dieses Netzwerk statt, um die Ausführung der Anwendung zu gewährleisten [vgl. IEC 61 499-1 2006, S. 18]#. Die IEC 61 499 bezeichnet die Funktionen, die spezifisch für die Lösung eines Problems sind und die ein System ausführt, als Anwendung oder Applikation [vgl. IEC 61 499-1 2006, S. 9]#. Eine Anwendung kann beispielsweise aus einer Prozessdatenverarbeitung bestehen, die auf einem Gerät implementiert ist. Das Einlesen der Eingänge übernimmt ein anderes Gerät und die Ausgabe an den Prozess

erledigt ein weiteres Gerät [vgl. von Aspern 2009, S. 48]. Zusammengefasst definieren Wurmus und IEC 61 499 ein verteiltes Automatisierungssystem somit als eine Steuerungsarchitektur, in welcher der Steuerungscode auf verschiedene Steuerungen und intelligente Geräte (im Folgenden Knoten genannt) verteilt ist, die direkt miteinander kommunizieren, um ihre Aufgaben zu erfüllen.

Die vom Automatisierungssystem zu erfüllenden Aufgaben werden in einer verteilten Steuerungsarchitektur in Funktionen unterteilt, die den jeweiligen Knoten zur Ausführung zugeordnet werden [vgl. Felser 2003, S. 15]. Somit können verteilte Automatisierungssysteme mit verteilter Steuerintelligenz realisiert werden. Der Vorteil der verteilten Steuerungsarchitektur gegenüber der zentralen und dezentralen ist der hohe Grad an Modularität und Flexibilität [vgl. Frank et al. 2012a, S. 294]*.

Typischerweise besteht ein verteiltes Automatisierungssystem aus verschiedenen Elementen, die im Nachfolgenden erklärt werden [Frank et al. 2012a, S. 294]*:

I/O-Geräte: Diese Geräte (Sensoren/Aktoren) bieten die Möglichkeit, mit dem gesteuerten Prozess zu interagieren, bieten aber keine Rechenressourcen.

Knoten I: Diese Knoten (zum Beispiel SPS ohne direkte Prozessschnittstelle) haben keine direkte Schnittstelle zum kontrollierenden Prozess. Die Knoten haben lediglich Rechenressourcen und die Möglichkeit, mit den I/O-Geräten zu kommunizieren.

Knoten II: Diese Knoten (zum Beispiel I/O-Geräte mit Rechenressourcen) haben eine direkte Schnittstelle zum kontrollierten Prozess. Des Weiteren bieten die Knoten Rechenressourcen sowie die Fähigkeit, mit dem gesteuerten Prozess zu interagieren.

Infrastrukturelemente: Diese Elemente (zum Beispiel Gateways, Bridges und Repeater) bieten die Infrastruktur, um eine Kommunikation zwischen den Elementen zu ermöglichen. Obwohl die Geräte nicht an der Umsetzung des Steuerungscodes beteiligt sind, sind sie ausschlaggebend für die Erfüllung von Anforderungen an verteilte Steuerungsanwendungen (beispielsweise Zeitanforderungen).

Die verteilte Steuerungsarchitektur, unabhängig von der Anbindung der Feldgeräte, ist in Abbildung 2.3 dargestellt.

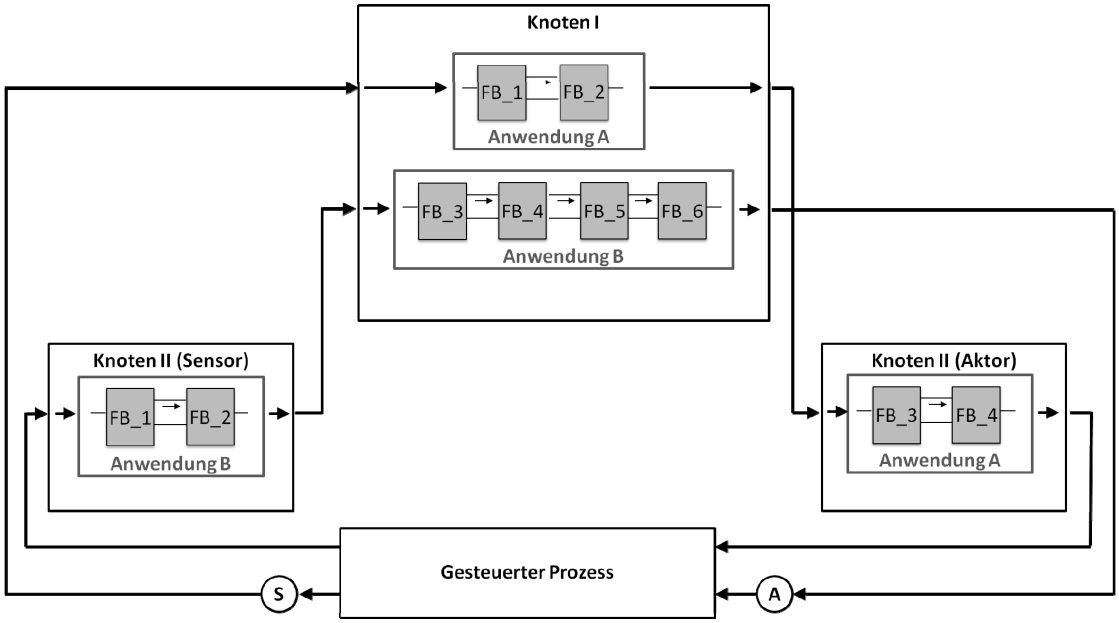

Abbildung 2.3: Verteilte Steuerungsarchitektur

Laut ISO/IEC 25010 [ISO/IEC 25010 2011][#] ist die hohe Verfügbarkeit einer verteilten Steuerungsarchitektur dann gegeben, wenn eine Komponente funktionsbereit ist, sobald sie benötigt wird [vgl. ISO/IEC 25010 2011, S. 13][#]. Durch den Ausfall eines Subsystems oder einzelner Komponenten ist das verteilte Gesamtsystem beeinträchtigt, aber es fällt nicht komplett aus. Die Teilanwendungen können im Idealfall auf andere Komponenten übertragen werden, sodass eine Fortführung des Prozesses ermöglicht wird. Dies ist in einem verteilten Automatisierungssystem einfacher realisierbar als in einer dezentralen Steuerungsarchitektur, da lediglich kleine Teile der Anwendung übertragen werden müssen und keine gesamte Anwendung. In einem verteilten Automatisierungssystem muss nicht jedes einzelne Signal an eine zentrale Steuerung geliefert werden, da innerhalb eines Knotens ein Signal verarbeitet werden kann. Die Vorteile von verteilten Automatisierungssystemen gegenüber dezentralen Systemen sind die Möglichkeit der Nutzung der Rechenleistung mehrerer Knoten für eine Anwendung sowie eine hohe Zuverlässigkeit der Systeme, da ein Ausfall einer Hardware-Komponente nicht zum Gesamtausfall beziehungsweise Teilausfall eines bestimmten Systems führt.

2.4 Zwischenfazit

Die Praxis hat gezeigt, dass zentrale Steuerungsarchitekturen nicht mehr den wachsenden Anforderungen der Anlagenbetreiber gerecht werden können [vgl. Feldhorst & Libert 2010, S. 29]. Die zentralen Steuerungsarchitekturen beinhalten große Softwaremodule, die aus »einem Guss« bestehen und bei denen Programmcodeänderungen bis in das »Herz« der Programmstruktur eingreifen müssen [vgl. Favre-Bulle 2004, S. 87]. Die damit verbundenen hohen Kosten bei Änderung oder Adaption der Anlage werden von den Kunden nicht mehr akzeptiert [vgl. Favre-Bulle 2004, S. 87]. Deshalb geht der Trend von zentralen Steuerungsarchitekturen mit großen »monolithischen« Softwareblöcken hin zu Architekturen mit verteilter Steuerintelligenz [vgl. Favre-Bulle 2004, S. 87]. In Abbildung 2.4 auf der nächsten Seite sind die drei Steuerungsarchitekturen (zentral, dezentral und verteilt) vergleichend dargestellt.

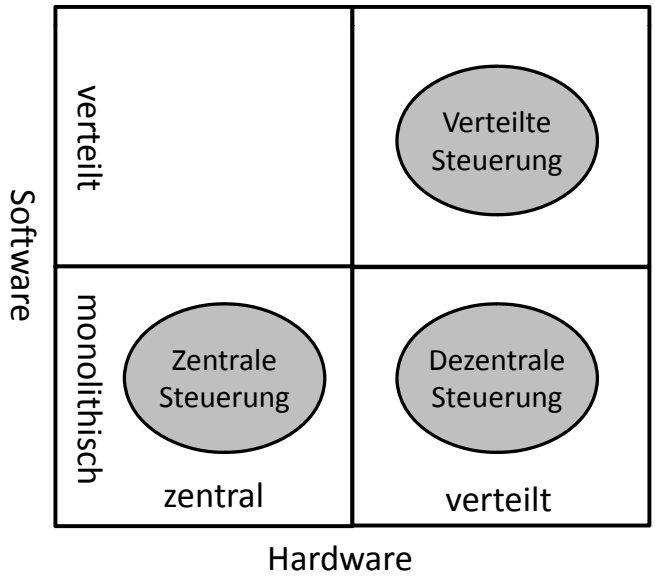

Abbildung 2.4: Vergleich der drei Steuerungsarchitekturen [vgl. Favre-Bulle 2004, S. 87]

In einer verteilten Steuerungsarchitektur ist die Steuerungssoftware in Funktionsblöcke unterteilt, die miteinander kooperieren, um eine gemeinsame Aufgabe zu erfüllen. Verteilte Steuerungsarchitekturen bieten dem Anwendungsentwickler, wie zuvor erläutert, eine Reihe von unterschiedlichen Verteilungsmöglichkeiten, insbesondere wenn komplexe Prozesse geregelt und gesteuert werden müssen. Diese komplexen Prozesse sind mit einer zentralen Steuerung nicht durchführbar, da die Rechenleistung des Systems nicht ausreichend ist.

Durch die zunehmende Komplexität in der Automatisierungstechnik und den vermehrten Einsatz immer anspruchsvollerer Funktionalität wird die Software der Automatisierungssysteme anfälliger für Störungen und Fehler [vgl. Felser 2003, S. 11], die sich oftmals auf die Arbeitsweise des Anwendungsentwickler zurückführen lassen. Aufgrund der zuvor beschriebenen Struktur verteilter Automatisierungssysteme ist der Umfang der Steuerungsaufgabe auf den einzelnen Knoten im Vergleich zur Gesamtautomatisierungsaufgabe gering [vgl. Papenfort 2006, S. 42]. Dies hat zur Folge, dass durch gezielte Verteilung der Steuerungsaufgaben beziehungsweise Automatisierungsfunktionen die Auswirkungen von Fehlern räumlich begrenzt sind und Konsequenzen reduziert werden sowie Fehler leichter zu finden sind. Des Weiteren hat die Modularität zur Folge, dass einzelne Komponenten unabhängig voneinander entwickelt, getestet und ausgetauscht werden können [vgl. Göpfert 2009, S. 123 f.]. Die Modularität unterstützt dadurch eine funktionale und physische Unabhängigkeit der Komponenten.

Die zunehmende Automatisierung sowie die steigende Funktionalität resultieren aus einer steigenden Komplexität, was eine wachsende Bedeutung für bestehende Anforderungen hat, die sich erhöhen oder verschärfen können [vgl. Papenfort 2006, S. 43] [vgl. Harbach et al. 2007, S. 262 f.]. Um die zunehmende Komplexität zu beherrschen, ist es hilfreich, eine verteilte Steuerungsarchitektur aufzubauen. Aufgrund der Leistungszuwächse im Bereich von Steuerungen und intelligenten Feldgeräten ist es möglich, Automatisierungsfunktionen gezielt verteilt auszuführen.

Ein verteiltes Automatisierungssystem benötigt mehrere Systemkomponenten, wie beispielsweise Hardware und Kommunikationsverbindungen, um die benötigte Automatisierungsaufgabe zu erbringen [vgl. Kurschl 2000, S. 38]. Die Verteilung der Steuerungssoftware muss so erfolgen,

dass trotz eines möglichen Ausfalls von Teilsystemen eine dauerhafte Verfügbarkeit des Systems sichergestellt werden kann [vgl. Kurschl 2000, S. 38]. Aufgrund der verteilten Steuerungsarchitektur können auch zeitkritische Abläufe sichergestellt werden, da die lokalen Knoten lediglich einen begrenzten Aufgabenbereich haben [vgl. Meier 2001, S. 11–16].

Aus der Sicht eines Anwendungsentwicklers besteht die Herausforderung beim Engineering von verteilten Automatisierungssystemen darin,

- dass eine Verteilung von Automatisierungsfunktionen auf unterschiedliche Knoten nötig ist, um die geforderten funktionalen Anforderungen unter Berücksichtigung der nfAs zu erfüllen,
- für die projektspezifischen funktionalen Anforderungen eine passende auf mehrere Steuerungen verteilbare Funktionsarchitektur zu finden und
- die Komplexität der Zusammenhänge von verteilten Funktionen und deren ausführender Hardware während des Entwurfs zu beherrschen.

Deshalb werden in Kapitel 3 unterschiedliche Strukturen beziehungsweise Vorgehensmodelle für verteilte Automatisierungssysteme analysiert. Des Weiteren wird die Einbindung von Anforderungen in das Engineering von Automatisierungssystemen bewertet.

3 Engineering von verteilten Automatisierungssystemen – Stand der Wissenschaft und Technik

In diesem Kapitel erfolgt eine Vorstellung des Stands der Wissenschaft und Technik im Bereich verteilter Automatisierungssysteme. Hierzu werden zum einen der Unterschied zwischen funktionalen und nicht-funktionalen Anforderungen beschrieben und zum anderen das Konzept der IEC 61 499. Darüber hinaus werden die verschiedenen Vorgehensmodelle und Strukturen verteilter Automatisierungssysteme vorgestellt sowie existierende Lösungsansätze, die sich mit Algorithmen für die Verteilung beschäftigen, besprochen.

3.1 Anforderungen im Engineering von verteilten Automatisierungssystemen

Dieser Abschnitt beschreibt die verschiedenen Arten von Anforderungen. Hierbei werden zuerst die funktionalen Anforderungen definiert und daraufhin die nicht-funktionalen. Im Anschluss wird die Relevanz von Anforderungen in der Automatisierungstechnik beschrieben.

3.1.1 Funktionale Anforderungen

Damit eine automatisierte Anlage ihren Zweck erfüllen kann, muss diese bestimmte funktionale Anforderungen erfüllen. Diese funktionalen Anforderungen beschreiben, was das System leisten soll, und sind Grundbestandteil eines jeden Systems [vgl. Vogel-Heuser et al. 2013, S. 4] [V-Modell xtb][@]. Zur Festlegung der funktionalen Anforderungen ergeben sich oftmals Fragestellungen, die beantwortet werden müssen. Diese Fragestellungen sind:

- Welche Funktionalität soll das Automatisierungssystem leisten?
- Welche Eingaben, Verarbeitungen und Ausgaben sind notwendig?
- Welches Verhalten soll in bestimmten Situationen praktiziert werden?
- Welches Verhalten soll in bestimmten Situationen nicht praktiziert werden?

Die funktionalen Anforderungen werden während der Anforderungsanalyse, zumeist in textueller Form, in einem Lasten- und Pflichtenheft [VDI 3694 2008][#] festgehalten. Diese Anforderungsanalyse ermöglicht ein gemeinsames Verständnis bezüglich der zu leistenden Funktionalität eines Systems. Funktionale Anforderungen sind Schritte eines technischen Prozesses, wie beispielsweise Flüssigkeit mischen, Flüssigkeit transportieren, Füllstand protokollieren und Werkstück bearbeiten.

3.1.2 Nicht-funktionale Anforderungen

Die nfAs beschreiben allgemeine Eigenschaften einer Anlage beziehungsweise eines Betriebsmittels [vgl. ISO/IEC 25 010 2011, S. 10 ff.][#] und beziehen sich auf die Qualität, in der die Funktionalität zu erbringen ist. Die nfAs können in herstellerneutrale Anforderungen sowie her-

stellerspezifische Anforderungen, wie beispielsweise bestimmte Technologien, unterteilt werden [vgl. VDI 3694 2008, S. 13][#]. Beispielhafte nfAs an ein System sind im Nachfolgenden aufgeführt:

- Nach Anschalten der Steuerung soll diese nach 10 s betriebsbereit sein.
- Der Füllstand soll für 10 Jahre archiviert werden.
- Der Signalton hat eine maximale Anschaltzeit von 2 ms.

Oftmals werden in Projekten lediglich die funktionalen Anforderungen betrachtet und die nfAs erst, wenn die Funktionalität nicht in dem gewünschten Maße erbracht werden kann (zum Beispiel schlechte Qualität, Prozessablauf zu langsam).

3.1.3 Einbindung von Anforderungen in die Automatisierungstechnik

Zu Beginn eines Engineering-Prozesses erfolgt eine Anforderungsanalyse. Diese Anforderungsanalyse ermittelt die funktionalen und nicht-funktionalen Anforderungen, die in einem Lastenheft festgehalten werden. Die VDI 3694 [VDI 3694 2008][#] zeigt eine mögliche Struktur eines Lastenhefts. Gerade im Bereich der Automatisierungstechnik erfolgt eine weitere Spezifizierung der funktionalen Anforderungen, wie beispielsweise in Form von R&I-Fließbild oder Stellenplänen [vgl. Holm et al. 2013, S. 4 ff.]. Die Verteilung sowie Implementierung der Software ist einer der letzten Arbeitsschritte während des Engineering-Prozesses [vgl. Holm et al. 2013, S. 9].

Den nfAs wird im allgemeinen Engineering-Prozess eine zumeist untergeordnete Bedeutung zugesprochen, da diese schwer spezifiziert werden können [vgl. Zou & Pavlovski 2006, S. 316]. Deshalb werden nfAs oftmals nicht ausreichend berücksichtigt. Dies ist darauf zurückzuführen, dass der Anlagenbetreiber verstärkt die Anlagenfunktionalität in den Fokus rückt und in der Planungsphase funktionsorientiert gearbeitet wird. Die nfAs, wie beispielsweise Zykluszeit, werden dann erst bei der Inbetriebnahme berücksichtigt, was oftmals zu spät ist. Ein Beispiel hierfür sind die nfAs Safety und Security, die im Engineering-Prozess oftmals zu spät betrachtet werden und ein nachträgliches Redesign des Systems zur Folge haben [vgl. Zou & Pavlovski 2006, S. 315]. Zur Lösung dieser Problematik existieren eine Vielzahl von Ansätzen (siehe Abschnitt 3.3 und Abschnitt 3.5), die eine Einbindung von nfAs in den Engineering-Prozess ermöglichen. Hierzu muss sowohl die Steuerungssoftware als auch deren Verteilung auf die Steuerungshardware den Anforderungen gerecht werden.

Der Beitrag [Zou & Pavlovski 2006] beschäftigt sich mit der Modellierung von Anforderungen in Systemarchitekturen. Die Autoren zeigen auf, dass es keinen Standard und keine Verfahren für die Modellierung von nfAs einer Systemarchitektur gibt und schlagen eine Erweiterung des »4 + 1 Sichtenkonzepts« in Form einer Steuerungssicht vor. Dadurch sollen die gesamten Anforderungen im gesamten Lebenszyklus dargestellt werden.

Aufgrund ihrer Struktur ermöglichen verschiedene Ansätze, wie beispielsweise Cyber-Physical Systems (cps), die Erfüllung von nfAs. Das Konzept der cps [Broy 2010] ermöglicht beispielsweise aufgrund seiner Struktur die Beachtung und Erfüllung der nfA Modularität. Verschiedene Forschungsansätze, wie beispielsweise in [Fuchs et al. 2012], [Fuchs & Vogel-Heuser 2012] und [Obst et al. 2013] beschrieben, erforschen die Modularität im Maschinen- und Anlagenbau.

3.2 IEC 61 499

Die industriellen Automatisierungsanlagen werden im Wesentlichen durch speicherprogrammierbare Steuerungen (SPSen) gesteuert, die aktuell in den Sprachen des Standards IEC 61 131-3 [IEC 61 131-3 2003]# programmiert werden. Die Grenzen der IEC 61 131 [IEC 61 131-3 2003]# bestehen darin, die Herausforderungen zu erfüllen, die bei verteilten Automatisierungssystemen entstehen [vgl. Basile et al. 2011, S. 621]. Eine Schwachstelle der IEC 61 131 ist die Unterstützung und Implementierung von verteilten Automatisierungssystemen, da diese nicht vorsieht, Anwendungen über mehrere Knoten zu verteilen.

Deshalb existierte die Notwendigkeit eines Standards zur Programmierung von verteilten Automatisierungssystemen. Hierzu wurde der Standard IEC 61 499 [IEC 61 499-1 2006]# ausgearbeitet und 2005 veröffentlicht. Die IEC 61 499 definiert eine Architektur und bietet Richtlinien zur Entwicklung sowie Modellierung von verteilten Steuerungs- und Regelungssystemen.

Die IEC 61 499 orientiert sich an einer Programmierung, basierend auf Funktionsblöcken, und stellt ein ereignisgesteuertes Architekturmodell dar [vgl. Favre-Bulle 2004, S. 91 f.]. Ein Funktionsblock (FB) ist die kleinste Komponente innerhalb des Modells und beinhaltet Algorithmen, die sein Verhalten definieren (siehe Abbildung 3.1). Diese Funktionsblöcke können als Bestandteil einer Anwendung im Automatisierungssystem verteilt werden [vgl. Neumann et al. 2000, S. 265]. Neben den Datenein- und -ausgängen sowie den internen Variablen enthalten Funktionsblöcke der IEC 61 499 auch Ereignisein- und -ausgänge (siehe Abbildung 3.1). Auf Basis einer Kombination von Funktionsblöcken, in der die Datenein- und -ausgänge sowie die Ereignisein- und -ausgänge miteinander verknüpft werden, kann die verteilte Anwendung programmiert werden [vgl. Favre-Bulle 2004, S. 93]. Der Aufbau eines Funktionsblocks nach IEC 61 499 ist in Abbildung 3.1 dargestellt.

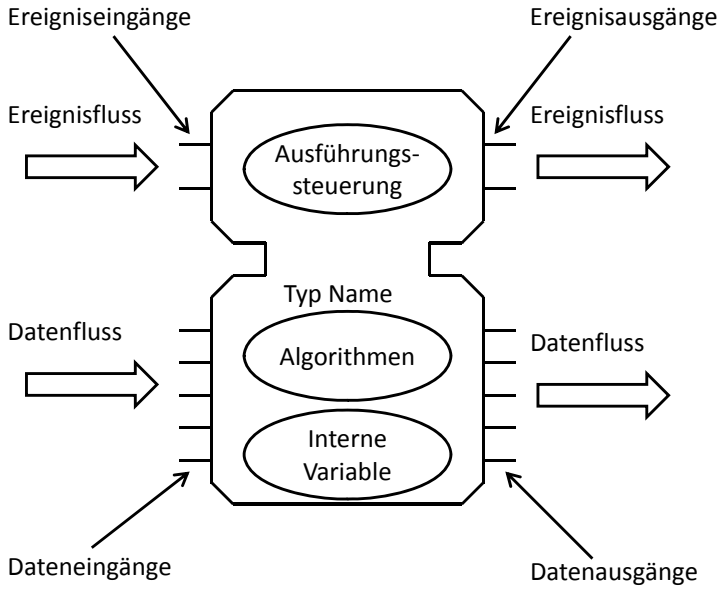

Abbildung 3.1: Aufbau eines Funktionsblocks nach IEC 61 499 [vgl. Favre-Bulle 2004, S. 92]

Die IEC 61 499-1 [IEC 61 499-1 2006]# definiert zum einen nicht nur Funktionsblöcke, sondern zum anderen unterscheidet sie auch zwischen verschiedenen Einheiten in einem Automatisierungssystem. Diese werden im Nachfolgenden erläutert und sind im Zusammenhang (ohne Anwendungsmodell) in Abbildung 3.2 beschrieben:

- Das *Systemmodell* beschreibt den physischen Aufbau eines Systems und beinhaltet alle Geräte sowie die sie verbindenden Kommunikationsmittel.
- Das *Gerätemodell* beschreibt die einzelnen Geräte, die mehrere Ressourcen beinhalten können. Diese Ressourcen sind funktionale Eigenschaften, wie beispielsweise Prozessoren oder Speicherplatz, eines Geräts [vgl. Neumann et al. 2000, S. 269]. Auf den Geräten laufen Anwendungen, entweder über mehrere Geräte verteilt oder nur auf einem.
- Im *Ressourcenmodell* werden die einzelnen Ressourcen der Geräte näher beschrieben. Die Ressource ist eine funktionale Einheit, die in einem Gerät enthalten ist und eine unabhängige Steuerung besitzt [vgl. IEC 61 499-1 2006, S. 19]#. In jeder Ressource können unabhängig Funktionsblöcke ausgeführt und gesteuert werden.
- Das *Anwendungsmodell* beschreibt das Anwendungsprogramm und besteht aus einem Netzwerk von Funktionsblöcken. Im Anwendungsmodell werden außerdem die Verbindungen zwischen den Funktionsblöcken definiert.

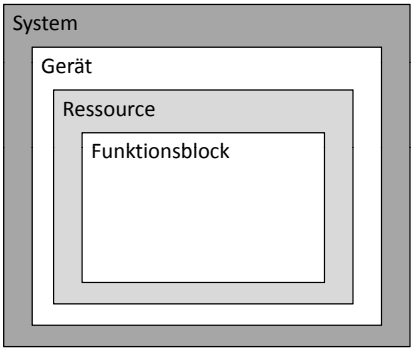

Abbildung 3.2: Zusammenhang der funktionalen Einheiten [vgl. Neumann et al. 2000, S. 270]

Für die Akzeptanz des Standards IEC 61 499 ist in der Industrie noch Forschungsarbeit notwendig, da zum einen die Semantik des Standards oftmals nicht ausreichend gefestigt ist und zum anderen weitere Modifizierungen und Erweiterungen am Modell erforderlich sind, um den gesamten Lebenszyklus eines Automatisierungssystems effektiv abzudecken [vgl. Thramboulidis 2006, S. 115]. Hierzu beschreibt Frey, dass durch die objektorientierte Erweiterung der IEC 61 131 kein Zusatznutzen bei Verwendung der IEC 61 499 in einem IEC 61 131-basierten Prozess entsteht und der Standard keine abstrakten Darstellungen oder neue Diagramme für die effiziente Modellierung weiterer Systemaspekte bereitstellt [vgl. Frey & Thramboulidis 2011, S. 7 f.]. Christensen geht noch weiter und trifft die Aussage, dass bei verteilten Anwendungen gerade dahinterstehende Ideen, wie beispielsweise Interface-Funktionsblöcke für die Anbindung der Anwendung an den Prozess sowie Event-getriebene Ausführungssteuerungen, noch nicht ausreichend erforscht wurden und argumentiert, dass diese neuen Konzepte durch Entwurfsmuster vereinfacht werden könnten [vgl. Christensen 2000, S. 63 ff.].

Frey untersuchte in [Frey & Thramboulidis 2011], ob die Einbindung des IEC 61 131-Standards in Kombination mit weiteren Ansätzen – wie beispielsweise IEC 61 499, Unified Modeling Language (uml) [uml]@ und Systems Modeling Language (sysml) [sysml]@ – zu einem abstrakten Entwurfsprozess im Sinne des modellgetriebenen Entwicklungsprozesses führt.

Aufgrund der geringen Akzeptanz der IEC 61 499, der objektorientierten Erweiterung der IEC 61 131-3 [IEC 61 131-3 2013]# sowie der grafischen Beschreibung von Anwendungen mithilfe der Funktionsbausteinsprache (FBS) stellt Frey heraus, dass der Einsatz von IEC 61 499 im modellgetriebenen Entwicklungsprozess keine Vorteile bringt.

Beim Engineering von verteilten Automatisierungssystemen nimmt die systematische Vorgehensweise sowie die Erfüllung von nfAs eine wichtige – ja sogar entscheidende Rolle – ein, da hiervon die Verteilung der Funktionsblöcke im Automatisierungssystem abhängt. Die IEC 61 499 unterstützt zum einen keine strukturierte Vorgehensweise zum Entwurf eines verteilten Automatisierungssystems und zum anderen auch keine Einbindung von nfAs in den Entwurfsprozess, bis auf wenige Ausnahmen, wie beispielsweise in [Sünder et al. 2006] für die nfAs Interoperabilität und Benutzbarkeit sowie in [Prayati et al. 2004] für die nfA Echtzeit beschrieben.

3.3 Vorgehensmodelle

Das erfolgreiche Engineering ist abhängig von einer strukturierten Herangehensweise sowie der Anpassung eines Vorgehensmodells und einer Struktur auf den jeweiligen Anwendungsfall. Ein Vorgehensmodell hat das Ziel, die Systementwicklung systematisch durchzuführen, um die Komplexität zu beherrschen. Die strukturierte Vorgehensweise wird zumeist in inhaltlich überschaubare Phasen eingeteilt, um eine inkrementelle Erstellung des Automatisierungssystems zu ermöglichen. Die einzelnen Phasen eines Vorgehensmodells können, je nach Modell, iterativ durchlaufen werden, sodass eine kontinuierliche Verfeinerung des Entwurfsschritts ermöglicht wird. Ein Vorgehensmodell stellt eine geordnete Herangehensweise bereit und strukturiert den Entwicklungsprozess. Dies ist gerade bei verteilten Automatisierungssystemen, die schwer zu entwerfen sind, hilfreich (siehe Abschnitt 2.4).

Für das Engineering von verteilten Automatisierungssystemen existiert kein spezielles Vorgehensmodell, das den Entwurfsprozess strukturiert. Die zuvor in Abschnitt 2.2.3 beschriebenen Vorgehensmodelle sind nicht ausreichend, da sie für verteilte Automatisierungssysteme nicht anwendbar sind oder aufgrund ihres Detaillierungsgrads schwer angepasst werden können. Deshalb besteht Forschungsbedarf hinsichtlich einer Vorgehensweise für den Entwurf verteilter Automatisierungssysteme, was auch Inhalt dieser Arbeit ist.

Einen ersten Ansatz im Hinblick auf ein Vorgehensmodell für verteilte Automatisierungssysteme beschreibt Hussain in [Hussain & Frey 2008]. Der dort dargestellte Entwicklungsprozess basiert auf UML-Diagrammen sowie FB-Diagrammen des Standards IEC 61 499 und berücksichtigt im vorletzten Schritt drei nfAs. Der Fokus dieses Vorgehensmodells liegt nicht auf der Verteilung der Funktionalität, sondern auf der Entwicklung von UML-Diagrammen im Entwurf verteilter Automatisierungssysteme und gibt keine Hilfestellung bei der Verteilung von Funktionalität. Im Schritt »Deployment« wird dargestellt, dass die Funktionsblöcke auf die Ressourcen verteilt werden sollen. In diesem Vorgehensmodell wurde der Schritt »Deployment« nicht in den Entwurfsprozess integriert, sondern als Schritt nach dem »Integrationstest« und somit nach dem Schritt »Implementierung«.

Eine weitere Vorgehensweise für verteilte Systeme beschreibt die Norm DIN 66 253-3 [DIN 66 253-3 1989]#. In dieser wird eine Vorgehensweise zum Aufbau eines verteilten Systems auf Basis der Programmiersprache *Process and Experiment Automation Realtime Language* (PEARL) beschrieben. Die DIN 66 253-3 beinhaltet eine Beschreibung der Stationen, der physikalischen Verbindungswege zwischen den Stationen, der Anschlüsse zur Peripherie und der Zuordnung

von Programmteilen zu Stationen [vgl. DIN 66 253-3 1989, S. 3][#]. Das Ziel der DIN 66 253-3 und der Entwicklung von PEARL war die Programmierung der in den 1970er und 1980er Jahren in der Anlagenautomatisierung eingesetzten Prozessrechnern [vgl. LAUBER & GÖHNER 1999, S. 301]. Mit der Entwicklung und industriellen Verbreitung von SPSen für die Anlagenautomatisierung verlor PEARL an Bedeutung und wird in industriellen Projekten nur noch selten eingesetzt [vgl. LAUBER & GÖHNER 1999, S. 301]. Die grundlegende Struktur, wie sie von PEARL in Bezug auf Hardware, Verbindungen, Ports und Verteilung von Software vorgegeben wird, wird heutzutage aber immer noch verwendet, da sich diese Struktur aufgrund ihrer Praxistauglichkeit bewährt hat.

PANJAITAN beschreibt in [PANJAITAN & FREY 2006] einen objektorientierten Modellierungsansatz für verteilte Automatisierungssysteme, basierend auf UML-Diagrammen und IEC 61 499. Der Ansatz umfasst insgesamt drei Schritte (FB-Level, System-Level, Implementation) und ist nicht mit einer strukturierten Vorgehensweise zu vergleichen, da lediglich erläutert wird, wie verteilte Automatisierungssysteme mit UML-Diagrammen dargestellt werden können. Des Weiteren beschreibt der Schritt *FB-Level* das interne Verhalten eines Funktionsblocks, was im weiteren Verlauf dieser Arbeit nicht betrachtet wird, da sie den Entwurf von verteilten Automatisierungssystemen zum Ziel hat und keine Implementierungsdetails aufzeigt.

Der Ansatz von PRAYATI, beschrieben in [PRAYATI ET AL. 2004], fokussiert auf den Entwurf von verteilten Automatisierungssystemen auf Basis der IEC 61 499 und beschreibt eine Vorgehensweise für heterogene Systeme. Hierzu werden Schritte definiert, die eine Zuordnung von Anwendungen auf Geräte ermöglichen. Der Ansatz von PRAYATI beinhaltet insgesamt sieben Schritte, wobei ein Rücksprung in vorherige Phasen nicht möglich ist. Die Nachteile hierbei sind, dass eine iterative Verteilung der Funktionalität nicht unterstützt wird und dass dieser Ansatz sehr stark an den Standard IEC 61 499 angelehnt ist, der in der Industrie noch keine Akzeptanz besitzt [vgl. THRAMBOULIDIS 2006, S. 115].

Die Ansätze von [PRAYATI ET AL. 2004] und [HUSSAIN & FREY 2008] beschreiben Vorgehensmodelle für verteilte Automatisierungssysteme, basierend auf dem Standard IEC 61 499, der noch Forschungsarbeit erfordert, um Akzeptanz in der Industrie zu erhalten [vgl. THRAMBOULIDIS 2006, S. 115].

3.4 Strukturen

Im Bereich verteilter Systeme existieren keine Vorgehensmodelle, sondern Strukturen, die die gesamte Architektur eines Systems beschreiben. Diese Strukturen sind mit einer strukturierten Vorgehensweise nicht zu vergleichen, da diese lediglich einen Anhaltspunkt geben, wie ein solches System physikalisch aufgebaut ist. Im Nachfolgenden werden die wichtigsten Strukturen verteilter Automatisierungssysteme vorgestellt.

AUTOSAR

Eine dieser Strukturen ist die AUTomotive Open System ARchitecture (AUTOSAR) [AUTOSAR][@], die ursprünglich für die Domäne des Automobils mit dem Ziel der Verteilung und Einbettung von Software auf unterschiedliche Steuergeräte im Fahrzeug entwickelt wurde. AUTOSAR ist eine Softwarearchitektur für Steuergeräte, in der die Software von der Hardware entkoppelt wird [vgl. DIEKHOFF 2010, S. 263 f.], und ist unterteilt in Schichten und Komponenten [vgl. EISEMANN ET AL. 2009, S. 51 ff.]. Die Software wird aus Modulen entwickelt (Softwarekomponenten)

und unterstützt somit die modulare Entwicklung von Steuerungssoftware [vgl. Frank et al. 2012a, S. 294]*. Zwischen den verschiedenen Schichten und zwischen den Modulen in den Schichten werden Schnittstellen spezifiziert [vgl. Diekhoff 2010, S. 263 f.]. Diese Spezifikation von Schnittstellen ermöglicht somit einen hohen Grad an Modularität und Konfigurierbarkeit [vgl. Peng et al. 2010, S. 189].

Die Architektur von AUTOSAR besteht aus einem Schichtenmodell mit sechs Schichten [vgl. Diekhoff 2010, S. 263 f.]. Die Grundidee dieser schichtenorientierten Architektur ist eine Unterteilung der Anwendungen in Komponenten sowie die Verteilung dieser Komponenten auf die Steuereinheiten. Hierzu müssen zum einen Kommunikations- und Datenaustauschbeziehungen beachtet werden und zum anderen die Speicherverwaltung. Der Standard AUTOSAR ist im Sinne einer strukturierten Herangehensweise kein Vorgehensmodell, da lediglich im Detail festgelegt wird, welche Daten und Schnittstellen in der Entwicklung beschrieben werden sollen und ein methodischer Ablauf bereitgestellt wird, der beschreibt, wie diese Daten verarbeitet werden [vgl. Eisemann et al. 2009, S. 51]. Standardisierte Architekturen, wie beispielsweise AUTOSAR, sind im automatisierungstechnischen Kontext eingebetteter Systeme, die wegen geringer Fertigungsdichte und für den flexiblen Einsatz in den unterschiedlichsten Anwendungen nicht mit standardisierten Plattformen ausgestattet werden können, ebenfalls technologieneutral. Die Vielzahl der Fahrzeugkomponenten erfordert eine ausgeprägte Spezialisierung, die im automatisierungstechnischen Umfeld aufgrund der Einzigartigkeit der Anlagen nicht hilfreich ist, um den Entwurfsprozess zu unterstützen.

SOA

Eine weitere Struktur für verteilte Automatisierungssysteme sind die Serviceorientierten Architekturen (SOA), die ursprünglich aus dem Bereich der Softwaretechnik kommen. Sie dienen dazu, verteilte Systeme einfach anzupassen. SOA beschreibt eine technologieunabhängige Architektur eines Systems [vgl. Feldhorst & Libert 2010, S. 33]. Die wichtigsten Bestandteile einer SOA sind Services, auch Dienste genannt, die in einer SOA lediglich aufgerufen und nicht in das System eingebettet werden. Diese Dienste repräsentieren Softwarekomponenten, die eine bestimmte Funktion eines Systems bereitstellen. Damit die Funktionalität eines Diensts genutzt werden kann, muss jeder Dienst über Schnittstellen verfügen [vgl. Feldhorst & Libert 2010, S. 34]. Eine SOA besteht aus unterschiedlichen Teilnehmern, wie beispielsweise Dienstanbieter, Dienstnutzer und Dienstvermittler [vgl. Melzer 2010, S. 16 f.] und beinhaltet Entwurfsprinzipien der Softwaretechnik, wie beispielsweise Kapselung, lose Kopplung und Verteilung [vgl. Feldhorst & Libert 2010, S. 34]. Der zuvor beschriebene Ansatz einer SOA könnte in die Automatisierungstechnik übertragen werden, was im Projekt »SOCRADES – Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded deviceS –«, beschrieben in [Bangemann et al. 2008] und [Bangemann et al. 2012], untersucht wurde. Hierbei wurde aufgezeigt, dass klassische Konzepte aus dem Bereich der IT, wie beispielsweise SOA, ohne Anpassungen nicht übernommen werden können [vgl. Bangemann et al. 2012, S. 3], da an die Automatisierungstechnik andere Anforderungen gestellt werden. Für den Einsatz von SOA über mehrere Ebenen der Automatisierungstechnik hinweg müssen grundlegende Paradigmen überdacht werden [vgl. Bangemann et al. 2012, S. 7]. Beispielsweise stellen Sensoren Daten und Informationen als *Services* der Umwelt zur Verfügung, während Aktoren ihre Funktionalität auch als *Services* bereitstellen [vgl. Bangemann et al. 2012, S. 7]. Im Gegensatz zu bestehenden Automatisierungssystemen müssen diese Services in einer SOA mit der Umge-

bung interagieren, das eigene Verhalten planen und anpassen sowie neue Verhaltensweisen und -strategien erlernen und optimieren [vgl. Bangemann et al. 2012, S. 7]. Diese Aussagen zeigen auf, dass die Einbindung von SOA in die Automatisierungstechnik möglich ist, jedoch noch viel Forschungsarbeit erfordert.

Agenten

Die Struktur der Softwareagenten gewinnt im Bereich der Automatisierungstechnik zunehmend an Bedeutung [Göhner et al. 2003] [Pech & Göhner 2011] [Frank et al. 2013b]. Die Richtlinie VDI/VDE 2653-1 [VDI/VDE 2653-1 2010][#] definiert einen Agenten als eine abgrenzbare Hard- oder Softwareeinheit mit einem definierten Ziel [vgl. VDI/VDE 2653-1 2010, S. 3][#]. Somit besitzt der Agent die Kontrolle über sein eigenes Verhalten und interagiert zum Erreichen der Ziele flexibel mit anderen Agenten und seiner Umgebung. Außerdem kann ein Agent selbstständig Entscheidungen treffen und danach handeln. Ein Agentensystem besteht aus einer Vielzahl von Agenten, die gemeinsam interagieren, um ein Aufgabe zu erfüllen [vgl. VDI/VDE 2653-1 2010, S. 3][#]. Der Einsatz des Agentenkonzepts wird in [Wagner & Göhner 2006] am Beispiel einer Produktionssteuerung sowie eines Energiemanagements aufgezeigt. Verschiedene Forschungsprojekte unterstützen die Berücksichtigung von bestimmten nfAs, wie [Folmer et al. 2012] am Beispiel der nfA *Echtzeit* aufzeigt.

Cyber-Physical Systems

Das Konzept der Cyber-Physical Systems (CPS), das ursprünglich aus dem Bereich der Softwaretechnik kommt [Broy 2010], wird in ersten Ansätzen auch auf die Automatisierungstechnik übertragen. Hierzu beschreibt [Vogel-Heuser & Witsch 2011] den Forschungsbedarf im Hinblick auf die Anwendung von CPS in der Automatisierungstechnik. Dieser Forschungsbedarf bezieht sich auf die Planung, den Entwurf sowie die Implementierung neuer Systeme. CPS sind aus Sicht der Informatik eingebettete Systeme (als Teil von Gebäuden, Produktionsanlagen, Logistikprozessen und Geräten), mit dem Ziel, über weltweite Dienste Anwendungsgebiete miteinander zu vernetzen [vgl. Broy 2010, S. 21 ff.]. Diese Systeme wirken mithilfe von Sensoren und Aktoren auf physikalische Vorgänge ein und erfassen physikalische Daten [vgl. Broy 2010, S. 21 ff.]. CPS nutzen weltweit verfügbare Dienste sowie Daten und verfügen über Mensch-Maschine-Schnittstellen, wie beispielsweise Browser. Wird diese neuartige Lösung aus der Informatik auf die Automatisierungstechnik übertragen, kann, auf Grundlage der von [Broy 2010] beschriebenen Spezifika, ein CPS mit einem verteilten Automatisierungssystem verglichen werden, das unternehmensübergreifend mithilfe von globalen Vernetzungen auf weltweit zur Verfügung stehende Dienste zugreifen kann. Dabei kommunizieren die Dienste über unterschiedliche Kommunikationsnetze miteinander, um ein gemeinsames Ziel zu verfolgen. Mehrere Konzepte und Anwendungsgebiete für CPS wurden in den letzten Jahren aufgezeigt und konzipiert, wie beispielsweise in [Vogel-Heuser & Witsch 2011], [Vogel-Heuser & Bayrak 2012] und [Industrie 4.0][@] beschrieben. Diese Konzepte zeigen auf, dass die Einführung des Paradigma CPS eine komplexe, langwierige Aufgabe ist und es noch einiger Forschungsarbeit bedarf.

Diskussion der beschriebenen Strukturen

Die zuvor beschriebenen Strukturen unterstützen den Anwendungsentwickler nicht bei der Aufgabe, ein verteiltes Automatisierungssystem unter Berücksichtigung von funktionalen und nicht-funktionalen Anforderungen sowie den Hardwareeinschränkungen zu entwerfen oder berücksichtigen nur einzelne nfAs, wie beispielsweise Zeitanforderungen. Die zuvor erläuterten

Strukturen verteilter Automatisierungssysteme beschreiben die gesamte Architektur eines Systems. Diese Strukturen sind mit einer strukturierten Vorgehensweise nicht zu vergleichen, da sie lediglich einen Anhaltspunkt geben, wie das System aufgebaut ist. Diese Architekturen unterstützen den Anwendungsentwickler üblicherweise nicht bei der Aufgabe, ein verteiltes Automatisierungssystem unter Berücksichtigung von funktionalen und nicht-funktionalen Anforderungen zu entwerfen oder berücksichtigen nur einzelne nfAs, wie beispielsweise Zeitanforderungen. Der Forschungsansatz der Agenten-Technologie findet im nachfolgenden Konzept keine weitere Berücksichtigung, da keine selbstorganisierenden Systemstrukturen beachtet werden. Es finden Automatisierungssysteme Berücksichtigung, bei denen die Verteilung der Funktionalität und die Kommunikationsbeziehung zum Planungszeitpunkt festgelegt werden.

3.5 Bewertung von Lösungsansätzen für das Problem der Verteilung

Die steigende Komplexität von Automatisierungssystemen macht es unabdingbar, verteilte Steuerungsarchitekturen aufzubauen, in denen der Steuerungscode auf mehrere Knoten verteilt wird, die direkt miteinander kommunizieren. In einer verteilten Steuerungsarchitektur kommen eine Vielzahl von Verteilungsmöglichkeiten für die Steuerungssoftware in Betracht. Insbesondere durch die wachsende Anzahl an Knoten und Automatisierungsaufgaben und der hieraus resultierenden Verteilung verändern sich auch immer mehr die nfAs, wie beispielsweise Zeitverhalten, Ressourcennutzung und Fehlertoleranz. Eine Möglichkeit, eine Steuerungsarchitektur für verteilte Automatisierungssysteme zu ermitteln, sind Algorithmen, die folgendermaßen unterschieden werden können:

1. Algorithmen, die ein Optimierungsproblem vollständig lösen wollen (globales Optimum beziehungsweise Pareto-optimal). Diese klassischen Algorithmen garantieren eine optimale Lösung.
2. Algorithmen, die mit heuristischen Methoden versuchen (begrenztem Wissen und geringer Zeitressource), eine gute, aber möglicherweise nicht optimale Lösung, für ein bestimmtes Problem zu erreichen.

Forschungsansätze, die das Verteilungsproblem vollständig lösen wollen, sind beispielsweise in [Wang & Chang 2008] beschrieben. Der Ansatz von [Wang & Chang 2008] beschreibt den Entwurf eines Netzwerks mithilfe einer Mehrfachoptimierung. Diese Mehrfachoptimierung findet bei *Pareto* seine optimale Lösung. Leider muss eine solche Lösung nicht immer die optimale Lösung für ein Netzwerkproblem sein, da sie abhängig von der Qualität der Optimierungskriterien ist.

Das vollständige Lösen des Optimierungsproblems ist zeit- und ressourcenintensiv. Deshalb beschreiben verschiedene Ansätze das Optimierungsproblem als ein Problem, das nicht vollständig gelöst werden kann, und wählen wiederum genetische Algorithmen für die Suche nach einer Lösung, die dem Optimum am nächsten kommen (heuristische Verfahren).

Der Entwurfsansatz von [Fencl et al. 2011] beschreibt den Entwurf eines verteilten Automatisierungssystems mithilfe eines genetischen Algorithmus. Die Autoren erläutern, dass es keinen Algorithmus gibt, mit dem der Entwurf eines verteilten Automatisierungssystems geschlossen durchgeführt werden kann. Die existierenden Algorithmen berücksichtigen lediglich ein oder zwei nfAs und betrachten das Verteilungsproblem als ein monokriterielles Optimierungsproblem oder als eines mit zwei Kriterien (Pareto-optimale-Lösung) [vgl. Fencl et al. 2011, S. 1288 f.].

Der von [Fencl et al. 2011] vorgeschlagene Algorithmus berücksichtigt drei nfAs und betrachtet das Verteilungsproblem als ein NP-vollständiges Problem, das nicht vollständig optimal gelöst werden kann, sondern für das lediglich eine Lösung gefunden wird, die dem Optimum nahe kommt.

Auf Basis der IEC 61 499 stellt [Hussain & Frey 2008] ein Konzept zur Spezifikation von Anforderungen und Randbedingungen für die Verteilung von Softwarekomponenten sowie einen Verteilungsalgorithmus vor, der auf diese Informationen zurückgreift. Dieser Algorithmus arbeitet zweistufig. Zunächst werden mögliche Lösungen auf die Vereinbarkeit der Randbedingungen mit Softwarekomponenten und Hardwareressourcen geprüft. Die möglichen Kombinationen werden dann mit einer Zielfunktion bewertet, der die spezifizierten Echtzeitanforderungen zugrunde liegen, die jeder Task einhalten muss. Dieser Algorithmus berücksichtigt im vorletzten Schritt die nfAs Beschränkungen der Ressourcen, Abhängigkeiten zwischen Software-Komponenten und Echtzeit. Dieser Algorithmus sucht eine Lösung nahe am Optimum, ohne die Erreichbarkeit der optimalen Lösung zu garantieren.

Das Konzept von Zeller, in [Zeller & Prehofer 2013] beschrieben, beschäftigt sich mit der Modellierung eingebetteter Echtzeitsysteme für Automobile. Die Autoren beschreiben eine mathematische Modellierung verschiedener nfAs, wie beispielsweise Speicherbedarf und Zeiteinschränkungen. Hierzu wird mittels gerichteter Graphen ein Zusammenhang von Funktionen, Sensoren und Aktoren dargestellt. Den Funktionen wird je eine Steuerung zugewiesen, auf der diese implementiert werden. Die Kommunikation zwischen den einzelnen Komponenten (Steuerung, Sensoren und Aktoren) wird mithilfe von ungerichteten Graphen dargestellt. Sobald neue Funktionen in einem bestehenden System hinzugefügt werden, muss die Frage der Zuordnung von Funktionen beantwortet werden. Diese Problematik beschreiben die Autoren als Allokationsproblem. Dieses Allokationsproblem – auch Verteilungsproblem genannt – wird unter Berücksichtigung von nfAs mithilfe verschiedener Methoden gelöst, wie beispielsweise verschiedener heuristischer Suchalgorithmen. Dieser Algorithmus sucht eine Lösung nahe am Optimum, ohne die Erreichbarkeit der optimalen Lösung zu garantieren.

Für die Kommunikation in einem Netzwerk beschreibt das Konzept von [Dutta & Mitra 1993] eine Methode, wie Optimierungsmodelle und heuristisches Entwurfswissen kombiniert werden können, um Entwurfsentscheidungen im Kommunikationsnetzwerk zu treffen. Hierzu wird in [Dutta & Mitra 1993] ein System eingeführt, das mithilfe eines Algorithmus verschiedene Alternativen ermittelt.

NfAs in Algorithmen

Entwurfsansätze für verteilte Systeme sind ein zentrales Thema beim Umgang mit der Komplexität und ermöglichen zwar eine Reduzierung des Engineering-Aufwands, berücksichtigen aber keine nfAs, wie beispielsweise der Ansatz von [Altiparmak & Dengiz 2009]. Der Algorithmus von altiparmak beschreibt den Aufbau einer physikalischen Topologie, berücksichtigt aber keine nfAs [Altiparmak & Dengiz 2009]. Andere Algorithmen lösen die Frage des optimalen Routings unter begrenzten Kosten, wie beispielsweise [Lin & Yeh 2010]. Dieser Algorithmus ermöglicht ein Routing mit Berücksichtigung von Zeit- und Kosteneinschränkungen. Der in [Misra et al. 2009] vorgestellte Algorithmus beschreibt das Routing zwischen mehreren Pfaden unter Bandbreiten- und Verzögerungseinschränkungen. Andere Algorithmen bewerten die Zuverlässigkeit bestehender Systemstrukturen, wie beispielsweise [Altiparmak et al. 2009]. Algorithmen, die eine durchschnittliche Verzögerung und ein Mindestmaß an Fehlertoleranz gewährleisten, werden beispielsweise in [Szlachcic 2006] und [Ko et al. 1997] beschrieben.

Diese Algorithmen werden verwendet, wenn ein gleiches Mindestmaß an Zuverlässigkeit im gesamten System erforderlich ist. Andere nfAs werden in den Algorithmen von [Dutta & Mitra 1993] und [Kumar et al. 1998] betrachtet, die sich nicht auf die Zuverlässigkeit, sondern auf Preis und Verzögerungszeit beziehen.

3.6 Zusammenfassung und Bewertung des aktuellen Stands in Wissenschaft und Technik

Die zuvor erläuterten Ansätze (siehe Abschnitt 3.1.3 bis Abschnitt 3.5) beschreiben zwar den Entwurf verteilter Automatisierungssysteme, berücksichtigen aber nur einzelne nfAs bei der Verteilung der Steuerintelligenz. Aufgrund der komplexen Struktur verteilter Automatisierungssysteme ist die Berücksichtigung einzelner nfAs oftmals nicht ausreichend, da sich aufgrund der vernetzten Struktur bestimmte nfAs verschärfen beziehungsweise verändern. Deshalb müssen die Ansätze weitere, speziell für verteilte Systeme relevante nfAs, berücksichtigen. Die Möglichkeit der Einbindung weiterer nfAs in bestehende Konzepte ist gering. Die zuvor beschriebenen Algorithmen lösen die Entwurfsprobleme unter anderem nicht vollständig, weil sie sich nur auf einen Teil des Problems konzentrieren und zumeist wichtige nfAs nicht beachtet werden. Deshalb sind die in Abschnitt 3.5 vorgestellten Algorithmen nicht dafür geeignet, ein Optimierungsproblem unter Berücksichtigung mehrerer nfAs zu lösen, da dadurch die Komplexität des Algorithmus zunimmt und die nfAs zueinander in Beziehung gesetzt und gewichtet werden müssen. Des Weiteren benötigen Suchalgorithmen verstärkte Rechenressourcen.

Der im Rahmen dieser Arbeit verfolgte Ansatz zur Unterstützung des Anwendungsentwicklers und zur Verteilung von Steuerungssoftware auf die Knoten basiert deshalb auf der Wiederverwendung bewährter, früherer Verteilungslösungen. Die Wiederverwendung ist ein wichtiger Ansatz im Entwurf verteilter Automatisierungssysteme und muss sowohl die funktionalen als auch die nicht-funktionalen Anforderungen berücksichtigen. Damit soll die Herausforderung einer Verteilung der Steuerungssoftware auf unterschiedliche Knoten unter Berücksichtigung von nfAs erleichtert werden. Durch die Wiederverwendung kann der Aufwand zur Integration der Lösung reduziert werden, da dieser im Vergleich zu einer Neuentwicklung geringer ist. Des Weiteren werden aufwändige Optimierungen durch die Wiederverwendung bewährter beziehungsweise getesteter Lösungen im Engineering reduziert. Hierzu müssen für jede Problemstellung geeignete Lösungen bereitgestellt werden. In dieser Arbeit wird ein Konzept vorgestellt, das die Grundstruktur zur Erstellung eines Entwurfsmusters aufgezeigt, nicht die individuellen Lösungen für jede Problemstellung.

Die Notwendigkeit von Wiederverwendung wurde auch in der Automatisierungstechnik erkannt und in verschiedenen Forschungsarbeiten behandelt, wie beispielsweise in [Witsch et al. 2008] und [Maga & Jazdi 2010] beschrieben. Deshalb befasst sich das nachfolgende Kapitel mit der Thematik Wiederverwendung. Ziel von Kapitel 4 ist die Ermittlung einer geeigneten Wiederverwendungsmethode für den Entwurf eines verteilten Automatisierungssystems.

4 Entwurfsmuster als Methode für Wiederverwendung im Engineering – Stand der Wissenschaft und Technik

Nachdem in Kapitel 2 die Grundlagen des Engineerings von Automatisierungssystemen und in Kapitel 3 der Stand der Wissenschaft und Technik zu verteilten Automatisierungssystemen dargestellt wurden, wird im Folgenden eine Möglichkeit zur Entwurfsunterstützung analysiert. Hierzu werden zum einen die Wiederverwendung sowie deren Grundlagen beschrieben und zum anderen wird die Wiederverwendungsmethode *Entwurfsmuster* vorgestellt.

4.1 Wiederverwendung im Engineering von verteilten Automatisierungssystemen

Im folgenden Abschnitt erfolgt eine Vorstellung des Stands der Wissenschaft und Technik im Bereich Wiederverwendung. Hierzu wird zunächst der Begriff Wiederverwendung erläutert und dann dessen Nutzen. Abschließend erfolgt eine Beschreibung der Wiederverwendungsmethoden sowie eine Bewertung des Einsatzes von Wiederverwendung in der Automatisierungstechnik.

4.1.1 Begriffsklärung Wiederverwendung

Aufgrund der zunehmenden Komplexität von Automatisierungssystemen und des vermehrten Einsatzes immer anspruchsvollerer Technik werden Automatisierungssysteme anfälliger für Störungen und Fehler. Diese lassen sich durch eine systematische Wiederverwendung von Teillösungen reduzieren. Die Wiederverwendung ermöglicht eine Reduktion von Kosten und Entwicklungszeit bei gleichzeitiger Steigerung von Qualität und Wirtschaftlichkeit [vgl. Alznauer et al. 2003, S. 31]. Die systematische Wiederverwendung von Teillösungen verfolgt das Ziel, bewährte Lösungen für eine Problemstellung in unterschiedlichen Automatisierungssystemen mit ähnlichen Problemstellungen einzusetzen [vgl. Fay et al. 2009, S. 83]. Die Grundidee der geplanten Wiederverwendbarkeit ist somit die Formalisierung beziehungsweise Strukturierung sich wiederholender Teillösungen. Deshalb führt Alznauer im Hinblick auf Wiederverwendung Folgendes aus: »Wiederverwendung ist daher der Schlüssel zu effizientem Engineering, zu einer sichereren, weil getesteten Anwendung im Betrieb und zu einer einfacheren Wartung.« [Alznauer et al. 2003, S. 32]

Im Gegensatz zur systematischen Wiederverwendung findet die ungeplante (Ad-Hoc) Wiederverwendung in fast jedem Engineering-Prozess eines Automatisierungssystems statt und zeichnet sich dadurch aus, dass keine Maßnahmen zur Systematisierung getroffen werden und die Wiederverwendung unkontrolliert, unkoordiniert sowie undokumentiert erfolgt [vgl. Rezagholi 1995, S. 39]. Ein Beispiel hierfür ist, dass bereits erstellte Lösungen »aus der Schublade gezogen« und auf aktuelle Problemstellungen angepasst werden. Da bei dieser Art der Wiederverwendung die alte Lösung schwierig an neue Anforderungen angepasst werden kann, ist der Erfolg stark begrenzt [vgl. Prieto-Díaz 1996, S. 6 f.]. Deshalb wird im Folgenden der Aspekt »systematische Wiederverwendung« weiterverfolgt. Hierzu ist es notwendig, wiederkehrende Problemstellungen während des Engineerings zu sammeln und zu klassifizieren, um allgemeingültige Lösungen für die Problemstellungen zu finden. Der Zeit- und Kostendruck, denen ein

Anwendungsentwickler ausgesetzt ist, hat zur Folge, dass im gesamten Engineering-Prozess versucht wird, bereits bestehende Automatisierungslösungen wiederzuverwenden. Der große Vorteil ist, dass die Grundidee zur Lösungserstellung bereits bearbeitet, zumeist schon geprüft wurde und sich bewährt hat. Dennoch muss die Grundidee der Automatisierungslösung auf die aktuelle Aufgabenstellung übertragen werden und kann ohne Änderungen nicht übernommen werden.

Im Kontext der Automatisierung konzentriert sich die Wiederverwendung auf Steuer- und Regelungsfunktionen, die trotz der Einzigartigkeit der Programmlogik und Implementierung von Automatisierungssystemen Gemeinsamkeiten besitzen, und deren Verteilung auf die Hardware [vgl. Alznauer et al. 2003, S. 31 f.]. Um der Wiederverwendung gerecht zu werden, müssen geeignete Automatisierungsfunktionen entwickelt und passgenau bereitgestellt werden, mit dem Ziel, durch eine Verknüpfung der Automatisierungsfunktionen sowie deren Verteilung die gewünschten Automatisierungslösungen zu realisieren. Die ISO/IEC 25 010 definiert die Wiederverwendbarkeit von Lösungen als den Grad, mit dem Automatisierungslösungen in mehr als einem System verwendet werden können oder wie Automatisierungslösungen zum Aufbau anderer Lösungen beitragen [vgl. ISO/IEC 25 010 2011, S. 15][#].

4.1.2 Nutzen der Wiederverwendung

Die Einführung von Wiederverwendung beinhaltet die Erwartungen, die Qualität zu steigern, Kosten zu reduzieren, Risiken und Zeitbedarf zu minimieren sowie die Flexibilität und Qualität zu steigern [vgl. Alznauer et al. 2003, S. 31] [vgl. Fay et al. 2009, S. 83]. Beim Einsatz von Wiederverwendung sinkt die Gefahr von Fehlern und der Aufwand zur Einbindung dieser Lösungen ist im Vergleich zu Neuentwicklungen geringer. Durch den Einsatz von Wiederverwendung entstehen laut Stützle [vgl. Stützle 2002, S. 32 f.] drei unterschiedliche Arten von Nutzen:

Reduzierter Aufwand für die Entwicklung und Wartung: Diese Art des Nutzens beinhaltet grundsätzlich zwei Ausprägungen. Dies sind zum einen *Senkung der Kosten* und zum anderen *Verkürzung der Entwicklungszeit.* Die Reduzierung des Aufwands ist auf die geringen Anteile an Neuentwicklung und die Wiederverwendung von Artefakten zurückzuführen.

Erhöhte Qualität der entwickelten Software: Diese Art des Nutzens besteht aus einer Vielzahl von Ausprägungen, wie beispielsweise *Nutzbarkeit für den Anwendungsentwickler*, *Portierbarkeit* und *Zuverlässigkeit für den Endnutzer.* Eine erhöhte Qualität der entwickelten Software kann dann erreicht werden, wenn die Software einen hohen Grad an Wiederverwendbarkeit aufweist.

Projektübergreifender Nutzen: Diese Eigenschaft des Nutzens beinhaltet beispielsweise die Ausprägung *Stärkung des Erfahrungsaustauschs der Entwickler.* Der Erfahrungsaustausch der Anwendungsentwickler hat den Vorteil, dass diese voneinander lernen und somit Fehler reduziert werden können.

Die Potentiale sowie der Nutzen der Wiederverwendung sind abhängig von deren Einsatzzeitpunkt, Einsatzspektrum sowie dem Anpassungsaufwand [vgl. Fay et al. 2009, S. 83].

4.1.3 Methoden der Wiederverwendung

Im Fachbereich der Softwaretechnik hegen Entwickler seit Jahrzehnten den Wunsch, Softwarebausteine wiederverwenden zu können und Anwendungen nicht individuell erstellen zu müssen. Deshalb haben sich in der Softwaretechnik unterschiedliche Wiederverwendungsmethoden be-

währt, wie beispielsweise Komponenten, Bibliotheken, Frameworks und Entwurfsmuster. Diese Wiederverwendungsmethoden unterscheiden sich zwar alle in der Anwendung, versuchen aber jeweils die Wiederverwendung von Softwarebausteinen zu ermöglichen.

Diese Ansätze der Wiederverwendung können auch auf die Automatisierungstechnik übertragen werden, da im Engineering von Automatisierungssystemen wie in der Softwaretechnik eine Lösung für eine individuelle, aber nicht neuartige Problemstellung gefunden werden muss. Deshalb werden im Folgenden die verschiedenen Methoden der Wiederverwendung im Engineering sowie deren Vor- und Nachteile erläutert.

Copy-Paste

Unter Wiederverwendung in Form von *Copy-Paste* wird das Kopieren von vorhandener Implementierung, das Einfügen in eine neu zu erstellende Lösung sowie die spätere Modifikation der kopierten Lösung an die Anforderungen des neuen Automatisierungssystems verstanden. Diese Form der Wiederverwendung führt jedoch zu einigen Problemen. Eine Systematisierung und Klassifizierung der Problemstellungen findet in dieser Form der Wiederverwendung nicht statt. Des Weiteren werden oftmals Vorlagen nicht korrekt angepasst, Fehler eingebaut oder eine fehlerhafte Implementierung übernommen ohne das notwendige Wissen über den Fehler. Diese Methode der Wiederverwendung unterstützt die Modifizierung des implementierten Codes nur eingeschränkt, da Änderungen in allen Systemen individuell eingearbeitet werden müssen.

Komponenten

Speziell in der Softwaretechnik hat sich eine Wiederverwendung in Form von *Komponenten* bewährt, die das Ziel hat, mithilfe von Softwarebausteinen vollständige Anwendungen (lauffähiges Programm) für Problemstellungen zu erstellen [vgl. Lindner 1996, S. 12]. Hierzu werden die einzelnen Komponenten ausgewählt, konfiguriert, parametriert und miteinander »verknüpft« [vgl. Dujmović 2002, S. 12]. Diese Form der Wiederverwendung hat die Vorteile, dass die Komponenten in verschiedenen Anwendungen verwendet und einfach ausgetauscht werden können sowie der Implementierungsaufwand gesenkt wird [vgl. Dujmović 2002, S. 12 f.]. Es ist zu beachten, dass Komponenten, die autonom voneinander entwickelt wurden, nicht ohne Probleme miteinander verknüpft werden können, und Anpassungen notwendig sind [vgl. Lindner 1996, S. 13]. Hierbei können Interoperabilitätsprobleme auftreten, da beispielsweise die kommunizierenden Komponenten gleiche Datentypen benötigen. Die Komponenten können abstrakt gehalten werden, was den Vorteil hat, dass sie universell einsetzbar sind. Der Nachteil hierbei ist aber, dass diese Komponenten miteinander »verknüpft« werden müssen, um eine Problemstellung zu lösen und der Anwendungsentwickler vor der Herausforderung steht, die richtigen Komponenten auszuwählen, um ein übergeordnetes Ziel zu erreichen.

Frameworks

Eine weitere Methode der Wiederverwendung in der Softwaretechnik sind die *Frameworks,* die einen »Anwendungsrahmen« bereitstellen. Ein Framework definiert somit die Architektur sowie den Kontrollfluss der Anwendung [vgl. Dujmović 2002, S. 17]. Diese generische Lösung muss vom Anwendungsentwickler an die jeweilige Problemstellung beziehungsweise den Anwendungsfall angepasst werden [vgl. Lindner 1996, S. 13]. Die Fehleranfälligkeit ist bei dieser Methode der Wiederverwendung stark reduziert, da die meisten Entwicklungsentscheidungen schon feststehen und der Anwendungsentwickler keinen Einfluss auf die Architektur der Anwendung nehmen kann [vgl. Dujmović 2002, S. 17]. Das Konzept der Frameworks basiert zumeist

auf dem Paradigma der Objektorientierung [vgl. Dujmović 2002, S. 17], da eine Standard-Softwarearchitektur definiert wird und nur an vorgegebenen Stellen anwendungsspezifischer Programmiercode beziehungsweise Logik eingefügt werden muss. Der Einsatz von Frameworks ist empfehlenswert, wenn die Anwendungen variabel in den Details sind und in unterschiedlichen Domänen Verwendung finden sollen [vgl. Dujmović 2002, S. 17]. Dadurch muss in Frameworks die Grundfunktionalität, die jede Anwendung besitzt, nicht mehrmals implementiert werden. Die so gewonnenen Ressourcen stehen für den anwendungsspezifischen Programmiercode zur Verfügung.

Entwurfsmuster

Eine weitere Wiederverwendungsmethode in der Softwaretechnik sind die *Entwurfsmuster*, die Lösungsvorschriften beziehungsweise Lösungsschablonen für ein komplexes sich wiederholendes Entwurfsproblem beschreiben. Ein Entwurfsmuster kennzeichnet somit eine allgemeingültige und wiederverwendbare Lösung für unterschiedliche Entwurfsprobleme und besteht aus einer Beschreibung des Entwurfsproblems, einer möglichen Lösung sowie den Randbedingungen, unter denen die Lösung gültig ist [vgl. Gamma et al. 2004, S. 3 f.]. Das Ziel, das mittels Entwurfsmuster verfolgt wird, ist, Anwendungsentwicklern zu helfen, Software schneller und besser zu entwerfen. Durch den Einsatz von Entwurfsmustern können zum einen Kosten und Risiken minimiert werden und zum anderen kann durch die Vorgabe von möglichen Lösungswegen sowie Designentscheidungen das Risiko von Fehlentwicklungen reduziert werden [vgl. Gamma et al. 2004, S. 3 f.].

4.1.4 Aktueller Einsatz von Wiederverwendung in der Automatisierungstechnik

Der Einsatz systematischer Wiederverwendung findet auch in der Automatisierungstechnik ein weites Anwendungsgebiet, weil beispielsweise die geplante Wiederverwendung zu einer Verbesserung der Software-Qualität führt. Laut Harbach werden unterschiedliche Themen, zu denen auch die Wiederverwendung von Entwurfsergebnissen gehört, zunehmend an Bedeutung gewinnen: »Themen wie die virtuelle Integration von heterogenen, vernetzten eingebetteten Systemen, die Wiederverwendung von Entwurfsergebnissen sowie die Analyse von funktionalen und nicht-funktionalen Anforderungen auf Modell- oder Softwarekomponenten-Ebene werden in den Mittelpunkt rücken.« [Harbach et al. 2007, S. 264] Dies zeigt auf, dass in der Automatisierungstechnik durch die Einführung neuer Paradigmen, wie beispielsweise Objektorientierung oder Softwareagenten, Forschungsbedarf hinsichtlich der Wiederverwendung besteht (siehe beispielsweise in [Fay et al. 2009] und [Thramboulidis 2008]). Diese Notwendigkeit von Wiederverwendung der Entwurfsergebnisse wurde erkannt, und mögliche Konzepte wurden in zahlreichen Forschungsarbeiten veröffentlicht. Einige Forschungsansätze für Wiederverwendung basieren auf der Framework-Technologie. Während die Konzepte von [Traub & Schraft 1999] und [Becker & Pereira 2002] ein objektorientiertes Framework für Echtzeitsysteme beschreiben, basieren andere Framework-Konzepte auf der IEC 61 499, wie beispielsweise [Cengic et al. 2006].

Cengic beschreibt in [Cengic et al. 2006] einen Ansatz zur Wiederverwendung in Form eines Frameworks für die Entwicklung einer komponentenbasierten verteilten Steuerungssoftware. Der Hauptfokus dieser Arbeit liegt auf der Wiederverwendungsmethode Komponenten. Cengic schlägt hierzu neue Softwarekomponenten, sogenannte Automatisierungskomponenten, vor, die hierarchisch eingebettet und verknüpft werden, um neue Komponenten zu entwickeln. Diese

Automatisierungskomponenten können auch kombiniert werden, um hierarchische komponentenbasierte Anwendungen zu entwickeln. Laut [Cengic et al. 2006] ist dieser Ansatz unabhängig von einer Ausführungsplattform, und Cengic zeigt die exemplarische Anwendbarkeit anhand der IEC 61 499. Der Schwachpunkt dieses Ansatzes ist die Verteilung der Komponenten, da lediglich ihre logische Verteilung auf einem Knoten beschrieben wird und nicht die räumliche Verteilung auf unterschiedliche Knoten. Des Weiteren betrachtet dieser Ansatz keine nfAs und gibt somit keine Hilfestellung für deren Erfüllung.

Der Forschungsansatz von [Dujmović 2002] beschreibt ebenfalls ein Konzept der Wiederverwendung, basierend auf der Framework-Technologie. Hierbei wird die Wiederverwendung von Implementierungscode sowie von domänenspezifischen Architekturen ermöglicht. Das Konzept besteht im Wesentlichen aus einem neu entwickelten Framework, dem Komponenten-Framework, sowie dessen Werkzeugunterstützung. Ziel des Konzepts ist es, den Anwendungsentwickler bei der Erstellung von Software zu unterstützen, ohne dass dieser selbst Programmierexperte sein muss. Die Integration in bestehende Entwicklungsprozesse kann vereinfacht werden, da dieses Konzept mit schon existierenden Ansätzen eingesetzt werden kann. Voraussetzung für die Anwendung dieses Konzepts ist eine klare Trennung von Funktionalität sowie Architektur, was im Bereich der verteilten Automatisierungssysteme nicht grundlegend erfolgen kann. Das Konzept unterstützt die Beachtung von nfAs in Form von Einschränkungen und Bedingungen, gibt aber keine Unterstützung zur Überprüfung deren Erfüllung.

Die zuvor beschriebenen Forschungsansätze zur Wiederverwendung in der Automatisierungstechnik beziehen sich auf Frameworks. Andere Konzepte lösen die Frage der Wiederverwendung mithilfe der Komponenten-Technologie, wie beispielsweise [Alznauer et al. 2003], [Eberle & Göhner 2003] und [Eberle & Göhner 2004]. Alznauer beschreibt in [Alznauer et al. 2003] eine Wiederverwendungsmethode, die auf Komponenten basiert und den Anwendungsentwickler mithilfe von Dialogen unterstützt. Diese Dialoge erlauben eine Instanziierung und ermöglichen eine Auswahl sinnvoller Varianten und eine Verknüpfung der Komponenten, die automatisch durchgeführt wird [vgl. Alznauer et al. 2003, S. 34]. Diese Methode hat den Vorteil, dass der Aufwand für Tests sowie die Variantenvielfalt reduziert werden, was eine Zeitersparnis zur Folge hat [vgl. Alznauer et al. 2003, S. 34]. Dieses Konzept unterstützt die Beachtung von nfAs nicht und gibt somit auch keine Hilfestellung für deren Erfüllung.

Eberle beschreibt in [Eberle & Göhner 2003] und [Eberle & Göhner 2004] eine komponentenbasierte Wiederverwendungsmethode für eingebettete Systeme. Hierzu beschreibt Eberle in [Eberle & Göhner 2003] charakteristische Merkmale einer Komponente, wie beispielsweise funktionale Geschlossenheit oder strukturelle Unabhängigkeit. Eberle erläutert, dass die Komponentenorientierung eine Fortentwicklung der Objektorientierung darstellt, da die Merkmale einer Komponente auch mit Objekten der Objektorientierung dargestellt werden können [vgl. Eberle & Göhner 2003, S. 52]. Auf Basis dieser grundlegenden Überlegungen der Komponentenorientierung beschreibt Eberle in [Eberle & Göhner 2004] ein Komponentenmodell, welches die charakteristischen Merkmale einer Komponente beinhaltet. Hierzu wird beschrieben, welche Notationen der Objektorientierung (uml) für die Modellierung komponentenbasierter Systeme verwendet werden können. Ziel des Ansatzes ist die strukturelle Unabhängigkeit der Komponenten. Dadurch soll eine Unabhängigkeit der Komponenten von der Applikation erreicht werden und eine Mehrfachverwendung von Systemteilen [vgl. Eberle & Göhner 2004, S. 72]. Dieses Konzept unterstützt die Beachtung von nfAs nicht und gibt somit auch keine Hilfestellung für deren Erfüllung.

Der Beitrag von [Maga & Jazdi 2010] beschreibt eine Klassifizierung von Beziehungen zwischen wiederverwendbaren Artefakten. Diese Beziehungstypen sind bei der Erstellung sowie dem Einsatz neuer wiederverwendbarer Artefakte relevant, da festgelegt wird, auf welche Art und Weise verschiedene wiederverwendbare Artefakte miteinander interagieren [vgl. Maga & Jazdi 2010, S. 347]. Hierzu beschreibt Maga die Aspekte *Domain Engineering, Domain Repository* und *Application Engineering*. Im *Domain Engineering* werden die wiederverwendbaren Lösungen entwickelt beziehungsweise erstellt, im *Domain Repository* werden die Lösungen gespeichert und das *Application Engineering* verwendet diese Lösungen, um ein neues Automatisierungssystem zu entwickeln [vgl. Maga & Jazdi 2010, S. 332]. Im Unterschied zu anderen Ansätzen betrachtet dieses Konzept das gesamte Automatisierungssystem unter dem Aspekt Wiederverwendung und nicht nur die Software [vgl. Maga & Jazdi 2010, S. 332]. Dieser Ansatz betrachtet keine nfAs und gibt somit keine Hilfestellung für deren Erfüllung.

Maga beschreibt in [Maga et al. 2011] Anforderungen an Engineering-Werkzeuge, um die Wiederverwendung zu steigern. Dazu werden Anforderungen an Entwickler von Engineering-Tools sowie Anforderungen an Entwickler von Automatisierungssystemen beschrieben. Maga beschreibt beispielsweise die Anforderung *Anpassung der Artefakte an das jeweilige Projekt*, die eine Steigerung von Wiederverwendung zum Ziel hat [vgl. Maga et al. 2011, S. 4]. Dies bedeutet, dass die wiederverwendbaren Artefakte abstrakt beschrieben sein müssen [vgl. Maga et al. 2011, S. 4]. Daraus lässt sich schließen, dass diese Artefakte unabhängig von eingesetzten Technologien sein müssen, also technologieneutral. Maga beschreibt zwar Anforderungen zur Erhöhung der Wiederverwendbarkeit und wie die Engineering-Werkzeuge und Artefakte angepasst werden müssen, zeigt aber keine konkreten Beispiele auf, wie solche Artefakte aussehen.

Der Forschungsansatz von [Witsch et al. 2008] beschreibt die Entwicklung von wiederverwendbarer Steuerungssoftware, basierend auf der Objektorientierung und uml. Hierzu werden Anforderungen an die Wiederverwendung mit Objektorientierung gestellt und das Konzept in CoDeSys 3[1] umgesetzt.

Aufgrund der wachsenden Komplexität und Flexibilität von Automatisierungssystemen gewinnt die Struktur der verteilten Automatisierungssysteme immer mehr an Bedeutung. Für den Anwendungsentwickler ist es durch die komplexen Strukturen von verteilten Steuerungsarchitekturen oftmals schwierig, eine geeignete Verteilung von Funktionalität vornehmen zu können. Deshalb sollte beim Entwurf einer verteilten Steuerungsarchitektur auf gute und bewährte Verteilungsmechanismen zurückgegriffen werden, um dadurch den Anwendungsentwickler zu unterstützen und die nfAs zu erfüllen. Die zuvor beschriebenen Konzepte zur Wiederverwendung unterstützen den Anwendungsentwickler beim Entwurf von Automatisierungssystemen. Diese Ansätze durch Komponenten oder Frameworks berücksichtigen zumeist keine Erfüllung von nfAs und betrachten auch nicht die Problemstellungen einer verteilten Steuerungsarchitektur.

4.1.5 Bewertung der Wiederverwendungskonzepte

Als Ergänzung zu den zuvor beschriebenen Wiederverwendungsmethoden muss analysiert werden, wann diese eingesetzt werden und welchen Nutzen diese Wiederverwendungsmethoden haben. Für die Wiederverwendung von bewährten Lösungen müssen diese bestimmte Merkmale und Eigenschaften erfüllen, um den Nutzen dieser Methoden zu messen. Eine Methode muss in unterschiedlichen Kontexten einsetzbar sein und somit die Eigenschaft »einsetzbar

[1] Die 3S-Smart Software Solutions GmbH ist Hersteller von CoDeSys, einem hardwareunabhängigen IEC 61 131-3 Programmiersystem zur Erstellung von Steuerungsanwendungen.

in unterschiedlichen Kontexten« erfüllen [vgl. Stützle 2002, S. 13]. Deshalb müssen die zuvor beschriebenen Wiederverwendungsmethoden anhand von Bewertungskriterien miteinander verglichen werden. Stützle definiert für die Wiederverwendung die Bewertungskriterien *Nutzbarkeit*, *Adaptierbarkeit* und *Portierbarkeit* [vgl. Stützle 2002, S. 13]. Im Folgenden werden diese Kriterien erläutert und auf die einzelnen Wiederverwendungsmethoden angewandt. Des Weiteren wird in Tabelle 4.1 auf der nächsten Seite dargestellt, in welchen Methoden die Kriterien erfüllt, teilweise erfüllt oder nicht erfüllt werden.

Nutzbarkeit

Das Kriterium *Nutzbarkeit* beschreibt den Aufwand, der für die Anwendung dieser Wiederverwendungsmethode notwendig ist. Die Nutzbarkeit beschreibt den Aufwand, der benötigt wird, um das Wiederverwendungskonzept zu verstehen und zu nutzen und auf dessen Basis Anwendungen zu entwickeln. Die *Nutzbarkeit* ist unterteilt in die Merkmale Verständlichkeit, Änderbarkeit sowie Verfügbarkeit [vgl. Stützle 2002, S. 14]. Das Merkmal Verständlichkeit beschreibt den Lernaufwand, um eine Wiederverwendungsmethode sowie deren Anwendung zu verstehen. Das Merkmal Änderbarkeit bewertet den Aufwand, der notwendig ist, um Änderungen an der Software umsetzten zu können. Das Merkmal Verfügbarkeit beschreibt den Aufwand, der notwendig ist, um von der Existenz wiederverwendbarer Software zu erfahren sowie die jeweils aktuelle Version zu erhalten [vgl. Stützle 2002, S. 14]. Diese drei Merkmale sind für den Anwendungsentwickler von immenser Bedeutung, da sie zum großen Teil über die Anwendung von Wiederverwendungsmethoden entscheiden. Deshalb muss der Aufwand für den Anwendungsentwickler überschaubar sein.

Bei der Wiederverwendung in Form von *Copy-Paste* ist die Nutzbarkeit des Konzepts eingeschränkt, da komplexe Quellcodezeilen ohne ausreichende Dokumentation schwierig oder kaum verständlich und Änderungen somit schwierig einzupflegen sind. Das Konzept der *Komponenten* ist relativ leicht zu verstehen und zu ändern, da der Schwerpunkt auf der Funktionalität sowie der Auswahl, Konfiguration und Parametrierung der Komponenten liegt. Das Konzept der *Frameworks* ist mit einem hohen Lernaufwand verbunden, da die gesamte Architektur des Frameworks verstanden werden und eine Vielzahl von Entwurfs- und Implementierungsentscheidungen getroffen werden muss, um das Konzept auf die jeweilige Problemstellung anzupassen [vgl. Dujmović 2002, S. 24]. Die Wiederverwendungsmethode des *Muster-Konzepts* ist aufgrund der übersichtlichen Darstellung – zumeist in der Modellierungssprache UML – leicht verständlich. Des Weiteren kann ein Muster leicht auf sich ändernde Bedürfnisse angepasst werden, da die Grundstruktur der Muster immer gleich ist. Dies bedeutet, dass auf Änderungen am Muster reagiert und die jeweils aktuelle Version des Musters bereitgestellt werden kann.

Adaptierbarkeit

Das Kriterium *Adaptierbarkeit* bewertet den Aufwand, der notwendig ist, um die Wiederverwendungsmethode an unterschiedliche funktionale und nicht-funktionale Anforderungen anzupassen. Stützle beschreibt, dass die Adaptierbarkeit ein wichtiges Merkmal ist, da die Wiederverwendungsmethode in verschiedenen Kontexten einsetzbar sein soll [vgl. Stützle 2002, S. 15]. Die *Adaptierbarkeit* muss sowohl die Funktionalität sowie die Fähigkeit beinhalten mit verschiedenen Systemen fachlich zusammenzuarbeiten und somit eine fachliche Systemintegration ermöglichen [vgl. Stützle 2002, S. 15]. Deshalb bewertet die *Adaptierbarkeit*, inwieweit der Anwendungsentwickler die Eigenschaften und das Verhalten der Wiederverwendungseinheit beeinflussen kann [vgl. Dujmović 2002, S. 25].

Die Konzepte *Copy-Paste* sowie *Komponenten* sind schwierig adaptierbar, da diese vorgefertigte und abgeschlossene Konstrukte darstellen und somit nur geringfügige Anpassungsmöglichkeiten bieten [vgl. Dujmović 2002, S. 25]. Die Konzepte *Entwurfsmuster* und *Frameworks* sind besonders gut adaptierbar, da diese flexibel an die funktionalen Anforderungen angepasst werden können [vgl. Dujmović 2002, S. 25]. Entwurfsmuster beispielsweise beschreiben Lösungsschablonen für ein komplexes sich wiederholendes Entwurfsproblem und können flexibel angepasst werden.

Portierbarkeit

Das Kriterium *Portierbarkeit* beschreibt die Fähigkeit der Wiederverwendungsmethode, in verschiedenen Umgebungen technisch einsetzbar zu sein und somit eine technische Systemintegration zu ermöglichen [vgl. Stützle 2002, S. 15]. Dieses Kriterium beinhaltet die Merkmale Installierbarkeit sowie Anpassbarkeit. Die Installierbarkeit beschreibt den Aufwand, der notwendig ist, um die Wiederverwendungsmethode in einer bestimmten Umgebung zu installieren. Die Anpassbarkeit beschreibt die Fähigkeit der Wiederverwendungsmethode, auf unterschiedliche Umgebungen angepasst zu werden.

Die Wiederverwendungen in Form von *Copy-Paste* und *Komponenten* sind schwierig portierbar, da diese abgeschlossene Konstrukte und Implementierungsdetails beinhalten und somit nur eine geringfügige Portierbarkeit in unterschiedliche technische Systeme ermöglichen. Die Konzepte *Entwurfsmuster* und *Frameworks* sind gut portierbar, da diese flexibel an technische Umgebungen angepasst werden können [vgl. Dujmović 2002, S. 25].

Zusammenfassende Bewertung

In Tabelle 4.1 erfolgt anhand der zuvor beschriebenen Kriterien eine abschließende Gegenüberstellung und Bewertung der zuvor beschriebenen Wiederverwendungskonzepte. Sie zeigt auf, dass die Konzepte der *Komponenten*, der *Frameworks* sowie der *Entwurfsmuster* eine gute Ausgangsbasis bilden, um den Anwendungsentwickler zu unterstützen. Aufgrund der negativen Bewertung des Konzepts *Copy-Paste* scheidet diese Form der Wiederverwendung aus.

Tabelle 4.1: Vergleich der Wiederverwendungskonzepte

	Copy-Paste	Komponenten	Frameworks	Entwurfsmuster
Nutzbarkeit	○	●	○	◐
Adaptierbarkeit	○	●	●	●
Portierbarkeit	○	○	●	●

● Kriterien erfüllt ◐ Kriterien teilweise erfüllt ○ Kriterien nicht erfüllt

Durch einen Vergleich der Konzepte *Frameworks* und *Entwurfsmuster* ist ersichtlich, dass beide Konzepte eine gute Ausgangsbasis für eine Entwurfsunterstützung des Anwendungsentwickler bereitstellen. Durch *Entwurfsmuster* wird diese Forderung noch weitreichender unterstützt, weil diese Wiederverwendungsmethode, im Gegensatz zu *Frameworks,* die Möglichkeit bietet, das Konzept leicht zu erlernen. Das Konzept der *Frameworks* ist nur schwierig umsetzbar, da seine Verständlichkeit durch eine Vielzahl von Entwurfs- und Implementierungsentscheidungen erschwert wird und die gesamte Architektur verstanden werden muss. Die *Komponenten* beinhalten abgeschlossene Konstrukte und Implementierungsdetails und ermöglichen nur eine

geringfügige Portierbarkeit in unterschiedliche technische Systeme. Deshalb sind diese beiden Konzepte zur Unterstützung des Anwendungsentwicklers bei der gezielten Verteilung der Funktionalität und der Erfüllung von nfAs nicht anwendbar. Eine weitere Möglichkeit, den Anwendungsentwickler bei der Verteilung der Funktionalität und der Erfüllung von nfAs mit Wiederverwendung zu unterstützen, ist die Wiederverwendungsmethode *Entwurfsmuster,* die im Nachfolgenden beschrieben wird.

4.2 Entwurfsmuster im Engineering von verteilten Automatisierungssystemen

Im nachfolgenden Abschnitt erfolgt eine Beschreibung des Stands der Wissenschaft und Technik im Bereich der Wiederverwendungsmethode *Entwurfsmuster.* Dazu wird zunächst der Begriff Entwurfsmuster erläutert sowie dessen aktuelle Historie. Anschließend erfolgt eine Bewertung des Einsatzes von Entwurfsmustern in den Disziplinen Informatik und Maschinenbau.

4.2.1 Historie der Entwurfsmuster

Der Begriff Muster wurde ursprünglich durch den Architekturprofessor Christopher Alexander für Architekturprobleme geprägt [Alexander 1964]. Alexander versuchte, komplexe Entwurfsprobleme im Bereich der Architektur in einfache, bekannte sowie lösbare Teilprobleme zu zerlegen. Das jeweilige Entwurfsproblem und dessen Lösungsvorschlag beschrieb er in einem sogenannten Pattern (Muster). Alexander definiert ein Muster wie folgt: »Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing the same way twice.« [Alexander et al. 1977, S. x] Den Durchbruch erlebte der Begriff aber vor allem im Bereich der objektorientierten Softwareentwicklung [Gamma et al. 2004]. In den 1990er Jahren wurde die Idee der Entwurfsmuster von der sogenannten »Gang of Four« (GoF) – Mitglieder sind Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides – aufgegriffen und auf die objektorientierte Programmierung übertragen, wo sie heute breite Anwendung findet [Gamma et al. 2004] [Lüder 2006].

4.2.2 Begriffserklärung Entwurfsmuster

Als eine Möglichkeit der systematischen Wiederverwendung haben sich in der Softwaretechnik Entwurfsmuster bewährt. Die Entwurfsmuster beschreiben eine allgemeine Vorgehensweise zur Lösung eines Entwurfsproblems und stellen allgemein anwendbare und wiederverwendbare Lösungsvorschläge bereit. Durch die Vorgabe von möglichen Lösungswegen wird das Risiko von Fehlentwicklungen reduziert. Des Weiteren werden diese Lösungsschablonen in Form von Entwurfsmustern genutzt, um Erfahrungen beim Entwurf von Software festzuhalten, mit dem Ziel, anderen Softwareentwicklern zu helfen, Software schneller und besser zu entwerfen. Die Anpassungen an funktionale Anforderungen, Aufgabenstellungen oder Entwurfsprobleme müssen vom Anwendungsentwickler durchgeführt werden. Ein Entwurfsmuster besteht nach [Gamma et al. 2004, S. 3 f.] aus den folgenden Elementen:

Mustername: Der Mustername ist ein Stichwort, um das Entwurfsproblem mit wenigen Worten zu beschreiben, und ist für ein gemeinsames Verständnis hilfreich.

Problembeschreibung: Die Problembeschreibung erläutert, wann das Muster angewandt werden kann, welches Problem behandelt wird und das Konzept des Entwurfsmusters. Des

Weiteren werden Entwurfsprobleme sowie Klassen- und Objektstrukturen beschrieben. Zusätzlich kann eine Liste von Bedingungen aufgestellt werden, die erfüllt sein müssen, um das Muster sinnvoll anzuwenden.

Lösungsbeschreibung: Dieser Abschnitt beschreibt die Elemente (Klassen oder Objekte), deren Beziehungen, Zuständigkeiten und Interaktionen. Es handelt sich hierbei um eine Art Schablone, wie ein Problem gelöst werden kann.

Konsequenzen: Hierbei werden die Vor- und Nachteile des Musters sowie mögliche Konsequenzen, die daraus resultieren können, beschrieben.

Die Entwurfsmuster der Softwaretechnik helfen dem Anwendungsentwickler bei der Lösung unterschiedlicher Problemstellungen, die beim Entwurf und bei der Implementierung von Softwaresystemen auftreten. Zum einen wird der Anwendungsentwickler beispielsweise bei der Auswahl passender Objekte unterstützt und zum anderen beim Bestimmen der Objektgranularität und beim Spezifizieren von Objektschnittstellen.

4.2.3 Entwurfsmuster in unterschiedlichen Disziplinen

Durch die unterschiedlichen Entwicklungen in den verschiedenen Wissenschaftsdisziplinen können Entwicklungsfortschritte anderer Wissenschaftsdisziplinen genutzt werden, um die eigene Entwicklung voranzutreiben. Deshalb werden im Folgenden die Einsatzgebiete von Entwurfsmustern in zwei Wissenschaftsdisziplinen betrachtet. Dies ist zum einen die Softwaretechnik, in der die Muster eine große Popularität erreicht haben, und zum anderen der Maschinenbau.

Entwurfsmuster in der Softwaretechnik

Die typischen Entwurfsmuster der Softwaretechnik lassen sich laut Gamma unterteilen in Erzeugungsmuster, Strukturmuster und Verhaltensmuster [Gamma et al. 2004]. Entwurfsmuster, die sich auf die Erzeugung von Objekten beziehen – *Erzeugungsmuster* –, »verstecken« den Erzeugungsprozess und machen ein System unabhängig davon, wie Objekte erzeugt, zusammengesetzt und repräsentiert werden [vgl. Gamma et al. 2004, S. 101]. Die klassenbasierten Erzeugungsmuster verlagern Teile der Objekterzeugung in Unterklassen und verwenden Vererbung, um die Klassen des Objekts zu variieren [vgl. Gamma et al. 2004, S. 101]. Die objektbasierten Erzeugungsmuster delegieren Teile der Objekterzeugung an andere Objekte [vgl. Gamma et al. 2004, S. 15].

Das *Strukturmuster* erklärt und verwaltet die Zusammensetzung von Klassen und Objekten [vgl. Gamma et al. 2004, S. 14 f.]. Um Schnittstellen und Implementierungen zusammenzufügen, verwenden die klassenbasierten Strukturmuster die Vererbung [vgl. Gamma et al. 2004, S. 169]. In den objektbasierten Strukturmustern werden Mittel und Wege beschrieben, um Objekte zusammenzufügen. Dadurch soll neue Funktionalität hinzugewonnen werden [vgl. Gamma et al. 2004, S. 169].

Die *Verhaltensmuster* beschreiben Algorithmen sowie die Zuweisung von Zuständigkeiten zu Objekten [vgl. Gamma et al. 2004, S. 271]. Des Weiteren werden in Verhaltensmustern die Interaktionen zwischen Objekten und Klassen beschrieben. Die klassenbasierten Verhaltensmuster verwenden Vererbung, um das Verhalten zwischen den Klassen zu verteilen, wohingegen die objektbasierten Verhaltensmuster Objektkomposition anstatt Vererbung verwenden [vgl. Gamma et al. 2004, S. 271].

Die Entwurfsmuster geben Lösungsvorschläge für bestimmte Entwurfsprobleme und beschreiben lediglich den logischen Aufbau eines Softwareproblems, die genaue Beschreibung des Codes bleibt offen (sprachenneutral). Laut der GoF sind die Entwurfsmuster »Beschreibungen zusammenhängender Objekte und Klassen, die maßgeschneidert sind, um ein allgemeines Entwurfsproblem in einem bestimmten Kontext zu lösen.« [Gamma et al. 2004, S. 4]. Aufbauend auf dem in den 1990er Jahren von der GoF entwickelten Musterkonzept, wurden mehrere Forschungsarbeiten zum Einsatz von Entwurfsmustern veröffentlicht. Der Einsatz dieser Entwurfsmuster gehört in der Softwaretechnik mittlerweile zum Stand der Technik.

In [Xu et al. 2006] wird ein Ansatz für Entwurfsmuster beschrieben, der eine getrennte Erfüllung von funktionalen und nicht-funktionalen Anforderungen in der Software ermöglichen soll. Dieser Ansatz ist beschränkt auf die objektorientierte Softwareentwicklung und beinhaltet keine Zeitanforderungen. Des Weiteren haben diese Entwurfsmuster keinen Bezug zu verteilten Systemen. Gross beschreibt in [Gross & Yu 2001] drei Entwurfsmuster, die einen starken Bezug zu nfAs haben. Der Schwerpunkt hierbei liegt auf den nfAs Zuverlässigkeit, Wartbarkeit und Entwicklungskosten. Der Ansatz stellt die Bezüge zwischen den nfAs und den funktionalen Anforderungen grafisch dar. Fletcher verfolgt in [Fletcher & Cleland 2006] einen ähnlichen Ansatz und ermöglicht – mithilfe von Entwurfsmustern – eine Beziehung zwischen Zielen und Lösungsansätzen herzustellen. Der Fokus liegt hier auf der nfA Sicherheit. Beide Ansätze zeigen auf, dass es möglich ist, einen Bezug zwischen nfAs und Lösung herzustellen, auch wenn diese Bezüge nur Tendenzen andeuten. Armoush beschreibt in [Armoush et al. 2009], wie die nfAs Sicherheit, Zuverlässigkeit, Wartbarkeit, Kosten und Befehls-Ausführungszeit in ein existierendes Entwurfsmusterkonzept integriert werden können. Hierzu schlägt Armoush für sicherheitskritische eingebettete Systeme eine Musterdarstellung vor. Die Konsequenzen auf die nfAs wurden darin eingebunden.

Entwurfsmuster in der Automatisierungstechnik

Entwurfsmuster fanden in den letzten Jahren in den unterschiedlichsten Einsatzgebieten weitere Verbreitung, einschließlich des Anwendungsgebiets der Automatisierungstechnik. Aufbauend auf den Entwurfsmustern der Softwaretechnik, wurden zahlreiche Forschungsarbeiten zur Konzeption sowie zum Einsatz von Entwurfsmustern veröffentlicht.

Eine der ersten Forschungsansätze zu Entwurfsmustern ist die Arbeit von Lea, beschrieben in [Lea 1994], in der mögliche Entwurfsmuster für die Luftfahrt erläutert werden. Lea beschreibt eine Architektur, in der die Problemstellungen eines Flugkontrollsystems in Teilprobleme zerlegt werden, die mithilfe von Entwurfsmustern gelöst werden können. In [Beder et al. 2000] wird als einer der ersten Ansätze ein Entwurfsmusterkonzept für die Automatisierungstechnik entwickelt, und zwar für das Teilproblem einer Eisenbahnsteuerung. Hierbei gehen die Autoren, wie sie selbst beschreiben, von unrealistischen Annahmen hinsichtlich Fehlerquellen und Fehlerauftreten aus. Ein weiterer Forschungsansatz, der sich mit Entwurfsmustern in der Steuerungstechnik befasst, ist die Arbeit von Xia, die in [Xia et al. 2003] beschrieben wird. Diese Arbeit beschreibt Entwurfsmuster für die Entwicklung von Steuerungssystemen auf Basis der IEC 61 499.

Die Arbeit von Douglass beschäftigt sich mit der Entwicklung von Entwurfsmustern für eingebettete Systeme [Douglass 2003]. Hierzu werden Entwurfsmuster beschrieben, die sich beispielsweise auf die Verteilung und Ressourcennutzung von eingebetteten Systemen beziehen. Die Entwurfsmuster der Ressourcennutzung definieren eine Ressource als ein Objekt, das dem Dienstnutzer verschiedene Services anbietet und Merkmale oder Eigenschaften besitzt [Douglass 2003, S. 205]. Douglass schreibt, dass eines der charakteristischen Merkmale

von eingebetteten Systemen die Verwaltung von Ressourcen ist. Die Ressourcenmuster von Douglass sollen helfen, diese Ressourcen zu organisieren, zu verwalten, zu nutzen und zu teilen. Die Verteilungsmuster von Douglass beziehen sich auf die Kommunikation sowie auf das Senden von Nachrichten. Es wird dargestellt, wie Informationen miteinander ausgetauscht und nicht wie Funktionalitäten verteilt werden sollen.

Der Ansatz von Fantuzzi beschreibt die Anwendung objektorientierter Modellierung auf komplexe Fertigungssysteme mit dem Schwerpunkt auf Lebensmittelproduktion [Fantuzzi et al. 2009]. Hierzu werden, basierend auf der UML-Modellierung, Entwurfsmuster eingeführt, die auf die Entwicklung von Maschinensoftware angewendet werden können. Das Entwurfsmusterkonzept von Fantuzzi betrachtet die nfA Echtzeit.

Ein Steuerungsentwurf, basierend auf Entwurfsmustern, wird in [Sanz & Zalewski 2003] vorgestellt. Diese Entwurfsmuster betrachten nicht nur das Problem der Software, sondern auch die domänenspezifischen Aspekte, wie beispielsweise eine Steuerungsstruktur. Sanz beschreibt ein Entwurfsmuster für die Erfassung von Messgrößen. Das Muster beschreibt den Ablauf der Messung sehr abstrakt, da lediglich dargestellt wird, dass eine Steuerung, ein kontrollierter Prozess, eine kontrollierte Variable sowie eine gemessene Variable benötigt werden und wie diese zusammenspielen. In den in Sanz vorgestellten Entwurfsmustern werden keine nfAs berücksichtigt.

In [Lüder et al. 2010] werden drei verschiedene Entwurfsmuster für verteilte Steuerungssysteme vorgestellt. Das dort beschriebene *Entwurfsmuster für verteilte Steuerungsanwendungen* ist unterteilt in Prozess- und Sicherheitsfunktionen. Diese Funktionen sind jeweils mit den Steuergeräten verknüpft und erlauben eine Kommunikation untereinander. Lüder beschreibt einen geräteunabhängigen Interaktionsmechanismus zwischen Funktionen ohne Kenntnisse darüber auf welchen Steuergeräten die Funktionen bereitgestellt werden. Lüder gibt keine Beispiele für Prozess- und Sicherheitsfunktionen und stellt auch keine konkreten Beispiele für eine Verteilung bereit. Er stellt lediglich dar, dass eine Funktion auf ein oder mehrere Steuergeräte verteilt werden kann und mit anderen Funktionen interagiert. Dieses Entwurfsmuster gibt aber keine konkrete Hilfestellung zur Zuordnung von Funktionen auf die Hardware.

Der Ansatz von Serna beschreibt das Handling von präventiven Instandhaltungen für Bearbeitungsmaschinen [Serna et al. 2011]. Hierzu wird ein Entwurfsmuster vorgeschlagen, das den fortschreitenden Verschleiß exponierter Maschinenteile unmittelbar nachweisen kann, bevor diese schadhaft werden. Dadurch können Instandhaltungsmaßnahmen rechtzeitig geplant werden, um weitere Probleme zu vermeiden. Serna beschreibt, wie die präventive Instandhaltung in einer Anwendungsumgebung in einer realen Steuerung umgesetzt werden kann. Hierzu müssen verschiedene Schritte durchgeführt werden, die in unterschiedliche Phasen (Entwicklungszeit, Anlagenbetrieb und Wartungsarbeiten) eingeteilt werden können. Die einzelnen Schritte der jeweiligen Phasen sind [Serna et al. 2011, S. 3 ff.] zu entnehmen. Die drei Phasen sind Teile des Entwurfsmusters und müssen bei dessen Anwendung durchgeführt werden. Die Anwendbarkeit dieses Entwurfsmusters für die Instandhaltung von Bearbeitungsmaschinen wird mithilfe der Cosme-Plattform (Machine Tool Distributed Control Platform For Communicating Machine Tools) [Catalan et al. 2011] dargestellt. Dieses Entwurfsmuster hatte zum Ziel, die Planung von Instandhaltungen zu vereinfachen und beinhaltet somit keine nfAs und bezieht sich auch nicht auf verteilte Steuerungsarchitekturen. Leider ist in diesem Ansatz nicht ersichtlich, inwieweit die Beschreibung des Entwurfsmusters auf den Grundlagen der Entwurfsmuster der Softwaretechnik basiert oder die Darstellung der Entwurfsmuster angepasst wurde.

Die Forschungsarbeit von Lüder beschreibt in [Lüder 2006] verschiedene Entwurfsmuster für agentenbasierte verteilte Steuerungssysteme. Hierzu wird das Entwurfsmuster *Entitäten in verteilten Auftragssystemen* vorgestellt, das festlegt, aus welchen Steuerungsbausteinen eine verteilte Auftragssteuerung bestehen soll. Das Entwurfsmuster *Agentenbasierte Steuerung von Produktionsaufträgen* beschreibt die Interaktionsmechanismen zwischen den Steuerungsbausteinen.

Das dritte Entwurfsmuster *Beschreibung von Produktion und Produktionsprozessen* wurde entwickelt, um zu beschreiben, welche Produktionsfunktion zur Ausführung eines Produktionsprozesses verwendet werden kann. Hierzu ist eine generische Beschreibung notwendig, die in diesem Entwurfsmuster ermöglicht wird. Das letzte Entwurfsmuster *Agentenbasierter Zugriff auf Produktionsfunktionen* beschreibt die Interaktion der Steuerungsbausteine mit den unterlagerten Steuerungsanwendungen auf Feldebene. Lüder beschreibt, dass für eine optimale Positionierung der Steuerungsbausteine im verteilten Steuerungssystem kein allgemeingültiges Entwurfsmuster entwickelt werden konnte, da sowohl mobile als auch stationäre Agenten eingesetzt werden können. Für diese Problemstellung werden jedoch zwei grundsätzliche Positionsvorschläge herausgearbeitet, die in [Lüder 2006] beschrieben werden. Der von Lüder vorgestellte Ansatz der Entwurfsmuster basiert stark auf dem Konzept der Agenten und beachtet keine nfAs. Dieser Ansatz orientiert sich bei der Darstellung der Entwurfsmuster an der Softwaretechnik.

Die zuvor beschriebenen Entwurfsmusterkonzepte der Informatik und der Automatisierungstechnik besitzen eine Vielzahl von Gemeinsamkeiten und versuchen den Anwendungsentwickler beim Entwurf eines Systems zu unterstützen. Die Darstellung der Entwurfsmuster orientiert sich in den bestehenden Konzepten des Maschinenbaus zumeist an der Darstellung der Informatik [vgl. Lüder 2006, S. 77 ff.] [vgl. Sanz & Zalewski 2003, S. 50 ff.] und beinhaltet lediglich minimale Änderungen. Im Gegensatz zu den Entwurfsmustern der Informatik erfolgt in den Entwurfsmustern der Automatisierungstechnik oftmals eine Einbindung der Hardware.

Die Art der Wiederverwendung ist bei den zuvor beschriebenen Ansätzen abhängig vom Einsatzspektrum. Während eine Vielzahl von Entwurfsmustern lediglich die funktionalen Anforderungen eines Systems beachtet, werden in einer kleinen Anzahl von Konzepten auch die nfAs betrachtet. Armoush beispielsweise betrachtet in [Armoush et al. 2009] die nfAs und beschreibt, wie diese eingebunden werden sollten, um herauszustellen, welche Auswirkungen dieses Entwurfsmuster auf bestimmte nfAs hat. Dieses Entwurfsmusterkonzept zeigt auf, dass es in Entwurfsmustern möglich ist, die Auswirkungen auf nfAs zu beschreiben und somit auch zu betrachten.

4.3 Zusammenfassung und Bewertung des aktuellen Stands in Wissenschaft und Technik

Der vermehrte Einsatz von verteilten Steuerungsarchitekturen hat zur Folge, dass mithilfe von leistungsfähigen Knoten die Automatisierungsfunktionen gezielt im System verteilt werden können. Die stets steigende Anzahl der Knoten und die damit einhergehenden Anforderungen an das Automatisierungssystem machen die Berücksichtigung und die Sicherstellung der Erfüllung von nfAs unabdingbar. Ein verteiltes Automatisierungssystem erfordert spezifische nfAs hinsichtlich seiner Komponenten, der Funktionsverteilung der Gesamtanwendung auf die Komponenten und der dazwischen stattfindenden Kommunikation. Beim Engineering von verteilten Automatisierungssystemen nimmt die Sicherstellung der Erfüllung von nfAs somit

eine wichtige und entscheidende Rolle ein und ist abhängig von der Funktionsverteilung. Im Fachbereich der Informatik, speziell im Teilgebiet der Softwaretechnik, laufen Forschungsvorhaben, die sich ebenfalls mit Möglichkeiten befassen, wie die Komplexität von großen Systemen im Entwurfsprozess praktikabel gestaltet werden kann. Ein in der Softwaretechnik weit verbreiteter Lösungsansatz für eine Entwurfsunterstützung sind die Entwurfsmuster, die technologieneutrale Musterlösungen für bestimmte Problemstellungen bereitstellen und sich dadurch auszeichnen, dass diese leicht anwendbar und verständlich sind. Dieser Ansatz der Entwurfsunterstützung durch Entwurfsmuster wurde in ersten Forschungsarbeiten (siehe Abschnitt 4.2.3) auf die Automatisierungstechnik übertragen.

Im Folgenden werden in den Tabellen 4.2 und 4.3 überblicksweise die in Abschnitt 4.2.3 vorgestellten wissenschaftlichen Arbeiten zu Entwurfsmustern benannt, soweit diese eine verteilte Steuerungsarchitektur und nfAs berücksichtigen. Des Weiteren wird bewertet, inwieweit der bewährte Aufbau der Entwurfsmuster, vorgeschlagen von der GoF (siehe Abschnitt 4.2.2), in den Arbeiten Anwendung findet. In den Tabellen wird dargestellt, in welchen Arbeiten die Kriterien erfüllt, teilweise erfüllt oder nicht erfüllt sind.

Tabelle 4.2: Arbeiten zu Entwurfsmustern mit Berücksichtigung von nfAs und verteilten Steuerungsarchitekturen – Bereich Softwaretechnik

	Entwurfsmuster in der Softwaretechnik		
Literatur	Bezug zu nfAs	Anwendungsbereich »verteilte Systeme«	Beschreibung der Muster bezogen auf die Informatik
[Gross & Yu 2001]	●	○	●
[Gamma et al. 2004]	○	○	●
[Xu et al. 2006]	◐	○	○
[Fletcher & Cleland 2006]	●	○	○
[Armoush et al. 2009]	●	◐	●

● Kriterien erfüllt ◐ Kriterien teilweise erfüllt ○ Kriterien nicht erfüllt

Tabelle 4.3: Arbeiten zu Entwurfsmustern mit Berücksichtigung von nfAs und verteilten Steuerungsarchitekturen – Bereich Automatisierungstechnik

	Entwurfsmuster in der Automatisierungstechnik		
Literatur	Bezug zu nfAs	Anwendungsbereich »verteilte Systeme«	Beschreibung der Muster bezogen auf die Informatik
[Beder et al. 2000]	◐	○	○
[Sandén 2001]	○	○	◐
[Grabmair et al. 2002]	◐	◐	●
[Xia et al. 2003]	○	●	○
[Douglass 2003]	○	◐	●
[Sanz & Zalewski 2003]	○	○	●
[Lüder 2006]	○	◐	●
[Brandl 2006]	○	◐	◐
[Fantuzzi et al. 2009]	◐	○	○
[Lüder et al. 2010]	○	●	○
[Serna et al. 2011]	○	○	○

● Kriterien erfüllt ◐ Kriterien teilweise erfüllt ○ Kriterien nicht erfüllt

Diese Einordnung der Arbeiten zu Entwurfsmustern in der Informatik und im Maschinenbau zeigt auf, dass sich die Entwurfsvorschläge zumeist auf zentrale und nicht auf verteilte Steuerungsarchitekturen beziehen. Des Weiteren wird in Tabelle 4.2 und 4.3 auf Seite 46 gezeigt, dass viele der Ansätze eine Berücksichtigung von nfAs nicht vorgesehen haben. Sofern die nfAs in den beschriebenen Ansätzen berücksichtigt werden, erfolgt dies zumeist textuell – bis auf wenige Ausnahmen –, sodass eine Erfüllung oder Nichterfüllung mit diesen Ansätzen schwierig nachvollziehbar ist.

Insgesamt zeigt diese explizit auf »Entwurfsmuster« fokussierende Forschungsrichtung, die aus der Informatik gespeist und vorangetrieben wird, auf, dass Entwurfsmuster geeignet sein könnten für die Erstellung von Software für verteilte Automatisierungssysteme und deren Verteilung unter Berücksichtigung von nfAs. Es wird auch aufgezeigt, dass dafür noch Forschungsarbeit erforderlich ist.

Anhand der in Kapitel 2 beschriebenen Randbedingungen und des in Kapitel 3 und Kapitel 4 analysierten Stands der Technik können im Folgenden Anforderungen an das zu entwickelnde Konzept formuliert werden. Das zu entwickelnde Konzept zum Entwurf verteilter Automatisierungssystem muss

- eine Vorgehensweise zum Entwurf von verteilten Automatisierungssystemen bereitstellen,
- unabhängig von der eingesetzten Technologie sein,
- den Anwendungsentwickler beim Entwurf unterstützen und
- nfAs im Entwurf berücksichtigen.

Auf Basis dieser Anforderungen wird im nachfolgenden Kapitel ein Konzept zum Entwurf verteilter Automatisierungssysteme entwickelt.

5 Herausforderung im Engineering von verteilten Automatisierungssystemen

Nachdem in Kapitel 2, Kapitel 3 und Kapitel 4 unter anderem die Grundlagen sowie der Stand der Wissenschaft und Technik bezüglich Steuerungsarchitekturen, Anforderungen, Vorgehensmodellen sowie Wiederverwendungsmethoden dargestellt wurden, erfolgt im nachfolgenden Kapitel eine abschließende Beurteilung des jeweiligen Stands der Technik zu *Vorgehensmodellen, Einbindung von Anforderungen* sowie *Wiederverwendung in Form von Entwurfsmustern.* Darauf aufbauend erfolgt in Abschnitt 5.4 eine Beschreibung der Konsequenzen für ein verbessertes Engineering in verteilten Automatisierungssystemen.

5.1 Vorgehensweisen beim Entwurf verteilter Automatisierungssysteme

In den Bereichen der Informatik und des Maschinenbaus existieren eine Vielzahl von Vorgehensmodellen, die im Engineering von Automatisierungssystemen Anwendung finden (siehe Abschnitt 2.2.3). Diese Vorgehensmodelle betrachten zumeist keine verteilte Steuerungsarchitektur und berücksichtigen somit nicht die speziellen Anforderungen eines verteilten Automatisierungssystems an

- seine Komponenten,
- die Funktionsverteilung der Gesamtanwendung auf die Komponenten und
- die dazwischen stattfindende Kommunikation.

Deshalb wird im Folgenden untersucht, ob die in Abschnitt 2.2.3 vorgestellten Vorgehensmodelle an die aktuelle Problemstellung angepasst werden können. Hierzu müssen die Vorgehensmodelle bestimmte Eigenschaften erfüllen.

Wiederverwendbarkeit

Um den Nutzen eines Vorgehensmodells für projektunabhängige Tätigkeiten vollumfänglich aufzuzeigen, ist der Aspekt seiner Wiederverwendung in vielen Kundenprojekten Voraussetzung [vgl. Fay et al. 2009, S. 83]. Die Wiederverwendung von Ergebnissen vorheriger Projekte führt zur Reduktion von Kosten und Entwicklungszeit und ist daher der Schlüssel zu einem effizienten Engineering [vgl. Alznauer et al. 2003, S. 31 f.]. Die »Wiederverwendung« bedeutet, dass die Vorgehensmodelle ihre vorgegebene Struktur in unterschiedlichen Projekten beibehalten müssen. Dadurch sind sie allgemeingültiger sowie einfacher nutzbar. Die in dieser Arbeit vorgenommene Klassifizierung in Bezug auf Wiederverwendbarkeit ist im Folgenden darstellt.

- ○ Das Vorgehensmodell unterstützt keine projektunabhängigen Phasen sowie Tätigkeiten und Wiederverwendung findet nicht statt.
- ◐ Das Vorgehensmodell unterstützt partiell projektunabhängige Phasen sowie Tätigkeiten und Wiederverwendung findet teilweise statt.
- ● Das Vorgehensmodell unterstützt projektunabhängige Phasen sowie Tätigkeiten und Wiederverwendung findet statt.

Anpassbarkeit

Laut Kuhrmann ist die Anpassbarkeit eine wichtige Eigenschaft eines Vorgehensmodells [vgl. Kuhrmann 2008, S. 23]. Im Kontext bedeutet dies, dass die Anpassung eines Vorgehensmodells auf konkrete Projektbedürfnisse sowie -anforderungen möglich sein muss. Die Anpassbarkeit kann laut Kuhrmann in zwei Ebenen unterteilt werden, und zwar in inhaltliche und strukturelle Anpassbarkeit [vgl. Kuhrmann 2008, S. 23]. Die inhaltliche Anpassbarkeit beschreibt die projektspezifischen Anpassungen, wie beispielsweise Hinzufügen oder Entfernen von Aktivitäten innerhalb einer Phase. Die strukturelle Anpassbarkeit bedeutet das Verändern existierender Strukturen des Vorgehensmodells, wie beispielsweise das Hinzufügen oder Entfernen ganzer Phasen. Die in dieser Arbeit vorgenommene Klassifizierung in Bezug auf Anpassbarkeit ist im Folgenden darstellt.

- ○ Das Vorgehensmodell unterstützt keine projektspezifischen Modifizierungen und Anpassbarkeit findet nicht statt.
- ◐ Das Vorgehensmodell unterstützt partiell projektspezifische Modifizierungen und Anpassbarkeit findet teilweise statt.
- ● Das Vorgehensmodell unterstützt projektspezifische Modifizierungen und Anpassbarkeit findet statt.

Komplexität

Die VDI 2206 beschreibt, dass die Beherrschung der Komplexität eine wichtige Aufgabe von Vorgehensmodellen ist [vgl. VDI 2206 2004, S. 23][#]. Speziell in automatisierten Anlagen, die durch eine Systemkomplexität gekennzeichnet sind, ist eine Beherrschung der Komplexität des Engineerings notwendig. Die Komplexität von Vorgehensmodellen beschreibt den Abstraktionsgrad sowie die mit ihm einhergehende Detailtiefe eines Vorgehensmodells. Der Abstraktionsgrad ist laut Kuhrmann eine wichtige Eigenschaft eines Vorgehensmodells [vgl. Kuhrmann 2008, S. 23]. Je umfassender und allgemeingültiger ein Vorgehensmodell beschrieben ist, desto ausgereifter ist es. Die in dieser Arbeit vorgenommene Klassifizierung in Bezug auf Komplexität ist im Folgenden darstellt.

- ○ Die im Vorgehensmodell dargestellten Phasen sind eingeschränkt beschrieben und eine Detaillierungstiefe ist nicht vorhanden oder das Vorgehensmodell besitzt eine Detaillierungstiefe und die Tätigkeiten sind nicht allgemeingültig beschrieben.
- ◐ Die im Vorgehensmodell dargestellten Phasen und Tätigkeiten sind partiell und allgemeingültig beschrieben.
- ● Die im Vorgehensmodell dargestellten Phasen und Tätigkeiten sind umfassend und allgemeingültig beschrieben.

In der nachfolgenden Tabelle 5.1 erfolgt anhand der zuvor dargestellten Kriterien eine abschließende Gegenüberstellung der in Abschnitt 2.2.3 beschriebenen Vorgehensmodelle. Die Kriterien sind entweder erfüllt, teilweise erfüllt oder nicht erfüllt.

Tabelle 5.1: Bewertung der Vorgehensmodelle

	Wiederverwendbarkeit	Anpassbarkeit	Komplexität
NA 35	◐	○	◐
VDI 2206	●	◐	◐
V-Modell	●	◐	○
V-Modell xt	◐	◐	●
3-Ebenen-Vorgehensmodell	◐	●	○

● Kriterien erfüllt ◐ Kriterien teilweise erfüllt ○ Kriterien nicht erfüllt

Die Gegenüberstellung der zuvor beschriebenen Vorgehensmodelle zeigt auf, dass diese unterschiedlich einzuordnen sind. Die NA 35 beinhaltet wenige Phasen und Tätigkeiten, die sich in Ihrem weiteren Verlauf wiederholen (Kostenschätzung). Der Methodenkatalog findet in sämtlichen Phasen Anwendung, was zum wiederholten Einsatz dieser Methode führt. Die NA 35 beschreibt detailliert, wie bei der Planung von leittechnischen Projekten vorzugehen ist. Dies bedeutet, dass die Phasen und Einzeltätigkeiten konkret festgelegt sind sowie wenig Spielraum bereitstellen. Das Vorgehensmodell NA 35 beschreibt detailliert das Vorgehen und ist partiell allgemeingültig beschrieben, weil die Tätigkeiten speziell an die Prozessleittechnik angepasst sind.

Die VDI 2206 beinhaltet Phasen und Tätigkeiten, die sich in ihrem weiteren Verlauf wiederholen. Dies geschieht zum Beispiel durch Iterationen. Die VDI 2206 erlaubt eine partielle Modifizierung des Vorgehensmodells. Die Phasen, Tätigkeiten und der Ablauf ermöglichen keine Anpassung. Der domänenspezifische Entwurf erlaubt jedoch, ein etabliertes domänenspezifisches Vorgehensmodell zu nutzen. Dadurch kann der Anwendungsentwickler den Inhalt der Entwicklungsmethodik auf das entsprechende Projekt anpassen. Die in der VDI 2206 definierten Phasen sind partiell allgemeingültig beschrieben. Die Phase des domänenspezifischen Entwurfs ist im Gegensatz zu den restlichen Phasen und Tätigkeiten nicht definiert.

Das V-Modell beinhaltet Phasen und Tätigkeiten, die sich in ihrem weiteren Verlauf wiederholen. Dies geschieht zum Beispiel durch Iterationen. Dieses Vorgehensmodell erlaubt projektspezifische Modifizierungen, weil die in den Phasen ablaufenden Tätigkeiten nicht beschrieben sind. Jedoch sieht die Struktur des Vorgehensmodells keine Anpassungen vor. Die im Vorgehensmodell dargestellten Phasen sind eingeschränkt beschrieben und eine Detaillierungsstufe ist nicht vorhanden.

Das V-Modell xt beinhaltet keine Phasen und Tätigkeiten, die sich in ihrem weiteren Verlauf wiederholen. Die Vorgehensbausteine finden in unterschiedlichen Phasen Anwendung, was zur wiederholten Anwendung dieser Vorgehensbausteine führt. Das V-Modell xt erlaubt eine partielle Modifizierung des Vorgehensmodells, abhängig von den in ihm definierten drei Projekttypen. Das Verändern dieser Projekttypen ist nicht vorgesehen. Die im V-Modell xt dargestellten Phasen und Tätigkeiten sind umfassend und allgemeingültig beschrieben.

Das 3-Ebenen-Vorgehensmodell beinhaltet Phasen und Tätigkeiten, die sich in ihrem weiteren Verlauf lediglich innerhalb einer Ebene wiederholen können. Das Vorgehen zur Umsetzung einer Komponente ist beim 3-Ebenen-Vorgehensmodell frei wählbar. Des Weiteren sind in keiner Phase explizite Aktivitäten definiert, sodass der Anwendungsentwickler flexibel entscheiden kann, welche Methoden er verwendet. Im 3-Ebenen-Vorgehensmodell sind die Phasen grob beschrieben sowie das Vorgehen.

Keines der zuvor beschriebenen Vorgehensmodelle genügt allen drei Kriterien. Somit ist mit diesen Vorgehensmodellen eine auf die Anforderungen verteilter Automatisierungssysteme angepasste Vorgehensweise schwierig. Aufgrund der weiten Verbreitung des V-Modells in der Industrie und der Bewertung in Tabelle 5.1 auf der vorherigen Seite, in der das V-Modell keine Detaillierungstiefe beinhaltet, könnte dieses so angepasst werden, dass die in Abschnitt 2.4 beschriebenen Anforderungen an verteilte Automatisierungssysteme berücksichtigt werden.

Somit reichen existierende Vorgehensmodelle und Strukturen des Maschinenbaus oder der Softwaretechnik, wie beispielsweise das V-Modell xt, das NAMUR-Arbeitsblatt 35 oder die Entwicklungsmethodik nach VDI 2206, nicht aus und sind aufgrund ihres Detaillierungsgrads schwierig anpassbar. Es fehlt somit ein Engineering-Ansatz, der den Anwendungsentwickler von den Anforderungen über den Entwurf bis zur Erstellung von verteilten Automatisierungssystemen unterstützt.

5.2 Integration von Anforderungen in den Entwurf

Der in Kapitel 3 und Kapitel 4 erläuterte Stand der Wissenschaft und Technik beschreibt verschiedene Methoden, mit denen bestimmte Lösungen für Teilprobleme des Entwurfsprozesses, wie beispielsweise Aufbau des Kommunikationssystems, Entwurf von Programmabläufen oder Auswahl von Komponenten, gelöst werden können. Zum einen sind diese Konzepte mathematisch komplex und können dadurch nur bedingt auf große Systeme übertragen werden und zum anderen erfordern diese Konzepte eine vollständige Spezifikation von Anforderungen, was in der Realität für große Systeme nicht gegeben ist.

Die Modellierung von Anforderungen hat im Bereich der Automatisierungstechnik bisher eine geringere Bedeutung als im Bereich der klassischen Softwaretechnik. Anhand eines Beispiels aus dem Anlagenbau untersuchte und spezifizierte Scherff in [Scherff et al. 1999] die frühen Phasen des Entwurfs und ermöglichte dem Anwendungsentwickler durch den Bezug von Anforderungen und projektierter Funktion eine anforderungsbasierte Projektierung.

Eine weitere Arbeit, die sich mit der Modellierung von Anforderungen in der Automatisierungstechnik beschäftigt, ist [Djambova 2005]. Djambova verwendet für die Beschreibung der Anforderungen verschiedene UML-Diagramme, wie beispielsweise das Anwendungsfalldiagramm mit Verfeinerung durch Aktivitätsdiagramme und Beschreibungsschablonen. Leider wird insbesondere der Zusammenhang zwischen nfAs und den dargestellten Beschreibungsschablonen unzureichend diskutiert und nicht ersichtlich.

Wie in Abschnitt 3.1.3 beschrieben, werden die Anforderungen üblicherweise im Rahmen der Lasten- oder Pflichtenhefterstellung in textueller Form festgehalten. In der Richtlinie VDI 3694 [VDI 3694 2008]# ist die Vorgehensweise für die Erstellung dieser Lasten- oder Pflichtenhefte beschrieben. Hierzu werden dort Hinweise in Form einer Checkliste gegeben. Die Anforderungen an das Automatisierungssystem werden direkt vom Kunden oder intern von vorgelagerten Gewerken und informell über firmeninterne Dokumentvorlagen formuliert. Anforderungen bei der Systementwicklung werden ebenfalls in unterschiedlichen Vorgehensmodellen berücksichtigt, wie beispielsweise in der Richtlinie VDI 2206 [VDI 2206 2004]#. Zumeist wird der Schritt der Anforderungserhebung in diesen Vorgehensmodellen aber als gegeben vorausgesetzt.

Einzelne der in Kapitel 3 beschriebenen Forschungsansätze, wie beispielsweise FMS [Herrero & Martinez 2010] sowie Software-Agenten [Folmer et al. 2012], ermöglichen aufgrund ihrer Struktur eine explizite Berücksichtigung von einzelnen nfAs, wie zum Beispiel Modularität

und Flexibilität. Gerade im Bereich Modularität existieren eine Vielzahl von Forschungsansätzen, wie beispielsweise [Obst et al. 2013] und [Fuchs & Vogel-Heuser 2012]. Fuchs und Schmitz, beschrieben in [Fuchs et al. 2012] und [Schmitz et al. 2013], definierten hierfür die Anforderungen aus dem Maschinen- und Anlagenbau. Diese Forschungsansätze unterstützen den Anwendungsentwickler üblicherweise nicht bei der Aufgabe, ein verteiltes System unter Berücksichtigung von funktionalen und nicht-funktionalen Anforderungen sowie den Hardwarebeschränkungen zu entwerfen oder berücksichtigen nur einzelne nfAs, wie beispielsweise Modularität [vgl. Frank et al. 2013a, S. 80]*.

In den existierenden Forschungsansätzen werden oftmals einzelne nfAs im Entwurfsprozess berücksichtigt (siehe Kapitel 4). Eine Berücksichtigung mehrerer nfAs erfolgt zumeist nicht, da die nfAs schwierig zu spezifizieren sind [vgl. Zou & Pavlovski 2006, S. 316]. Bestimmte nfAs, wie beispielsweise Safety oder Security, werden im Engineering-Prozess oftmals zu spät berücksichtigt, sodass ein aufwändiges Redesign des Systems erforderlich ist, um die nfAs zu erfüllen. Damit eine Nichterfüllung von nfAs erst überhaupt nicht zum Tragen kommt, wenn die Funktionalität nicht im gewünschten Maße erbracht wird, sollte den nfAs bereits in den Entwurfsphasen eine angemessene Rolle zugesprochen werden. In Kombination mit einer systematischen Vorgehensweise könnten in jeder Entwurfsphase, in der unterschiedliche Entwurfsschritte ausgeführt werden, nfAs hinterlegt werden, die in der jeweiligen Phase berücksichtigt und erfüllt werden müssen. Dadurch ist schon in einer frühen Phase des Entwurfsprozesses eine Nichterfüllung von nfAs erkennbar.

5.3 Integration von Wiederverwendung in den Entwurf

Die Integration von Wiederverwendung sowie funktionaler und nicht-funktionaler Anforderungen erlangt auch in der Automatisierungstechnik zunehmend Bedeutung [vgl. Harbach et al. 2007, S. 264] [vgl. Fay 2009, S. 52–55]. Die von Harbach in [Harbach et al. 2007] beschriebenen Schwerpunkte, wie beispielsweise vernetzte Systeme sowie Wiederverwendung von Entwurfsergebnissen, die eine Herausforderung im Bereich der Automatisierungstechnik darstellen, werden in dieser Arbeit zum Teil weiterverfolgt. Dem Ziel der Wiederverwendung sollte dadurch Rechnung getragen werden, dass mögliche Automatisierungslösungen verteilter Automatisierungssysteme in Form von Entwurfsmustern dargestellt werden, die einerseits auf nfAs referenzieren und andererseits eine geeignete Beschreibungsform verwenden [vgl. EIT Bericht 2010, S. 5 f.].

In der Praxis wird beim Entwurf und in der Realisierung von Automatisierungssystemen das Prinzip der Wiederverwendung für bestimmte Teilaufgaben schon lange verwendet, um die Komplexität von Automatisierungsanlagen zu beherrschen [vgl. Fay 2009, S. 53 f.]. Die Wiederverwendung von Artefakten bezieht sich zum einen auf materielle Artefakte, wie beispielsweise Motoren und Pumpen, und zum anderen auf immaterielle Artefakte, wie zum Beispiel Pläne, Architekturen, Funktionsbausteine und Spezifikationen [vgl. Fay 2009, S. 53 f.]. In der Verfahrenstechnik ist das Rohrleitungs- und Instrumenten-Fließbild (R&I-Fließbild) das zentrale Dokument, das die Anforderungen beinhaltet. Das R&I-Fließbild beinhaltet die physikalischen Anlagenelemente, wie beispielsweise Behälter, Rohrleitungen und Pumpen, sowie die automatisierungstechnischen Aufgaben, wie zum Beispiel Anzeigen, Schalten und Regeln. Des Weiteren beinhaltet das R&I-Fließbild Informationen über die Instrumentierung und ist mit Prozessleittechnik-Stellenblättern (PLT) sowie PLT-Stellenplänen verknüpft. Ein erfahrener Anwendungsentwickler erkennt anhand des R&I-Fließbilds bekannte Automatisierungsaufga-

ben und kann darauf aufbauend eine Lösung erarbeiten. Erste Ansätze für wiederkehrende Artefakte automatisierungstechnischer Lösungen in Form von Entwurfsmustern – ohne dass der Begriff Entwurfsmuster eingeführt wird – werden für Betriebsarten in [Frey 2004] aufgezeigt.

Aufgrund des in Kapitel 4 beschriebenen Stands der Wissenschaft und Technik besteht hinsichtlich einer systematischen Vorgehensweise sowie Entwurfsunterstützung und der damit verbundenen Einbindung von Wiederverwendung in den Entwurfsprozess von verteilten Automatisierungssystemen sowie der Berücksichtigung von nfAs noch hinreichend Forschungsbedarf.

In der Informatik, speziell im Teilgebiet der Softwaretechnik, wird an unterschiedlichen Möglichkeiten geforscht, die Komplexität von großen Systemen im Entwurfsprozess handhabbar zu machen. Ein in der Softwaretechnik weit verbreiteter Lösungsansatz für eine Entwurfsunterstützung sind die Entwurfsmuster, die technologieneutrale Musterlösungen für bestimmte Problemstellungen bereitstellen. Deren Vorteile wurden in Kapitel 4 näher erläutert und begründet.

5.4 Konsequenzen für ein verbessertes Engineering

Auf Grundlage der zuvor beschriebenen abschließenden Beurteilung des Stands der Wissenschaft und Technik erfolgt in den nachfolgenden Abschnitten eine Analyse des Handlungsbedarfs, auf dessen Basis das Konzept dieser Arbeit entwickelt wurde.

5.4.1 Handlungsbedarf

Die zunehmende Komplexität von Automatisierungssystemen, der vermehrte Einsatz immer anspruchsvollerer Technik und der Wunsch, diese Systeme gesamtheitlich optimal zu betreiben, führen zu einer Zunahme einer gezielten Verteilung von Regelungs- und Steuerungsaufgaben [vgl. EIT Bericht 2010, S. 5 f.]. Aufgrund anspruchsvollerer Technik, die anfälliger für Störungen und Fehler ist, und der räumlichen Ausdehnung der Systeme werden verteilte Steuerungsarchitekturen aufgebaut, in denen verschiedene Komponenten Automatisierungsaufgaben ausführen und mittels eines Kommunikationssystems miteinander vernetzt sind [vgl. EIT Bericht 2010, S. 5 f.]. Die Automatisierungssysteme sind oftmals komplex, müssen auf sich ändernde Anforderungen rasch reagieren und beinhalten eine hohe Anzahl an Steuerungen, Sensoren und Aktoren, die eine komplexe und stark vernetzte Struktur bilden. Der Entwurf einer verteilten Steuerungsarchitektur besteht aus einer Vielzahl von Herausforderungen, die durch eine Entwurfsmethodik wie folgt unterstützt werden müssen:

- Komplexität der Zusammenhänge von verteilten Funktionen und deren ausführender Hardware während des Entwurfs beherrschen
- Für projektspezifische funktionale Anforderungen eine passende auf mehreren Steuerungen verteilbare Funktionsarchitektur finden
- Verteilung von Automatisierungsfunktionen auf unterschiedliche Knoten, um die geforderten funktionalen Anforderungen unter Berücksichtigung der nfAs zu erfüllen

Es besteht Handlungsbedarf hinsichtlich einer systematischen Vorgehensweise für den Entwurf eines verteilten Automatisierungssystems. Die in den vorangegangenen Kapiteln diskutierten herkömmlichen Vorgehensmodelle fokussieren im Allgemeinen auf systematische Vorgehensweisen, zeigen aber keine konkrete Herangehensweise für verteilte Automatisierungssysteme

auf. Auch wenn es Vorgehensmodelle für verteilte Automatisierungssysteme gibt, wie beispielsweise in [Hussain & Frey 2008] und [DIN 66 253-3 1989]# beschrieben, beziehen sich diese auf bestimmte Technologien, wie beispielsweise IEC 61 499, und können nur mit erheblichem Aufwand angepasst werden. Eine systematische und technologieunabhängige Entwurfsmethode existiert allerdings noch nicht. Am Beispiel eines verteilten, eingebetteten Systems diskutiert der Beitrag [Broy et al. 1999] ein Konzept für die Einbindung von AutoFocus auf unterschiedlichen Abstraktionsebenen. DIN 66 253-3 [DIN 66 253-3 1989]# beschreibt eine grundsätzliche Vorgehensweise von PEARL-spezifischen verteilten Systemen, welche die Aspekte Hardware, Verbindungen, Ports und Verteilung von Software beinhalten. Ähnliche Vorgehensweisen wären wünschenswert, um ein verteiltes, technologieunabhängiges Automatisierungssystem zu entwerfen.

Ein verteiltes Automatisierungssystem stellt spezifische nfAs an seine Komponenten, die Funktionsverteilung der Gesamtanwendung auf die Komponenten und die dazwischen stattfindende Kommunikation. Insgesamt liegen die Schwerpunkte derzeitiger Forschungsansätze für einen anforderungsgetriebenen Steuerungsentwurf auf der Berücksichtigung und Erfüllung einzelner nfAs, wie beispielsweise in [Obst et al. 2013] beschrieben. Am Beispiel eines Verteilungsalgorithmus wird im Beitrag [Fencl et al. 2011] als einer der ersten Ansätze die Berücksichtigung und Erfüllung von mehreren nfAs diskutiert. Auch wenn es inzwischen erste Einbindungen von nfAs gibt, so fehlt es an einer Methodik für die systematische Einbindung der nfAs in den Entwurfsprozess verteilter Automatisierungssysteme [vgl. EIT Bericht 2010, S. 5 f.]. Dabei bedeutet »systematisch«, die besonderen nfAs, die sich aus der Verteilung ergeben, dann zu berücksichtigen, wenn diese im Entwurfsprozess relevant sind.

Weiterer Handlungsbedarf besteht hinsichtlich einer Wiederverwendung von Lösungen. Die in den vorherigen Kapiteln diskutierten Forschungsansätze bezüglich Wiederverwendung fokussieren im Allgemeinen auf zentralistische Strukturen und nicht auf die Anforderungen und Herausforderungen von verteilten Steuerungsarchitekturen. Auch wenn es inzwischen erste Ansätze für verteilte Automatisierungssysteme gibt, wie beispielsweise [Xia et al. 2003], so beziehen sich diese auf bestimmte Technologien, wie beispielsweise IEC 61 499, und lassen sich schwierig auf andere Technologien anpassen. Im Entwurfsprozess von Automatisierungssystemen für Produktionsanlagen wird Wiederverwendung seit zirka 1994 eingesetzt, wie beispielsweise in [Alznauer et al. 2003] beschrieben. Aufgrund der in Abschnitt 4.1.3 vorgenommenen abschließenden Beurteilung der unterschiedlichen Wiederverwendungsmethoden wird im nachfolgenden Konzept die Wiederverwendung in Form von *Entwurfsmustern* weiterverfolgt. Am Beispiel eines Entwurfsmusters wird im Beitrag [Armoush et al. 2009] ein Konzept für die Einbindung von nfAs in bestehende Entwurfsmusterkonzepte diskutiert. Eine ähnliche Vorgehensweise wäre für die Einbindung von nfAs in Entwurfsmustern für verteilte technologieunabhängige Automatisierungssysteme wünschenswert. Erste Forschungsarbeiten, die das Konzept der Entwurfsmuster auf die Automatisierungstechnik übertragen haben, sind charakteristisch dafür, dass

- kein expliziter Bezug zwischen den vorgeschlagenen Lösungen und den damit erfüllten nfAs hergestellt wird,
- die Lösungen auf die Implementierung bezogen sind und eine Lösung vorgehalten wird, nicht das Lösungsprinzip, die keinen Technologiewechsel überdauert und
- Aspekte der Verteilung von Automatisierungsaufgaben in verteilten Systemen nicht oder nur unzureichend behandelt werden.

Um die in der Informatik erfolgreich erforschten und eingesetzten Entwurfsmuster mit den Anforderungen von verteilten Automatisierungssystemen zu verbinden, besteht daher Forschungsbedarf hinsichtlich

- der Systematisierung von nfAs an ein verteiltes Automatisierungssystem,
- der Erstellung von Entwurfsmustern für verteilte Automatisierungssysteme und
- einer Darstellung der nfAs in den Entwurfsmustern, die einen Zusammenhang zwischen nfAs und den Lösungen herstellt, sodass über diese Zusammenhänge die Erfüllung von nfAs überprüfbar wird.

Die grundlegende Zielsetzung besteht darin, ein systematisches Vorgehensmodell für verteilte Automatisierungssysteme aufzuzeigen und das Vorgehen des Anwendungsentwicklers gezielt durch Entwurfsmuster zu unterstützen. Deshalb muss das nachfolgende Konzept

1. eine Vorgehensweise zum Entwurf von verteilten Automatisierungssystemen bereitstellen,
2. unabhängig von der eingesetzten Technologie sein,
3. den Anwendungsentwickler beim Entwurf unterstützen und
4. nfAs im Entwurf berücksichtigen.

5.4.2 Konzeptentwicklung

Das im Rahmen dieser Arbeit vorgestellte Konzept (siehe Abbildung 5.1) fokussiert auf eine Methodik für den systematischen Entwurf verteilter Automatisierungssysteme. Diese Entwurfsmethodik zielt auf eine systematische Vorgehensweise sowie eine Unterstützung des Entwurfsprozesses durch Entwurfsmuster und die Einbindung von nfAs in den Entwurfsprozess und in die Entwurfsmuster ab. Dadurch werden dem Anwendungsentwickler für die gewünschte und spezifizierte Funktionalität – abgeleitet aus funktionalen Anforderungen – jeweils bewährte typische Automatisierungslösungen – über »Entwurfsmuster« – zur Verfügung gestellt. Des Weiteren wird der Anwendungsentwickler in die Lage versetzt, die gewünschte Funktionalität so zu verteilen, dass nfAs erfüllt werden. Deshalb wird er beim Entwurf der Verteilung der Funktionalität dadurch unterstützt, dass ihm für die gewünschte und spezifizierte Funktionalität jeweils bewährte typische Verteilungsalternativen – ebenfalls über »Entwurfsmuster« – angeboten werden, welche die gewünschten nfAs erfüllen.

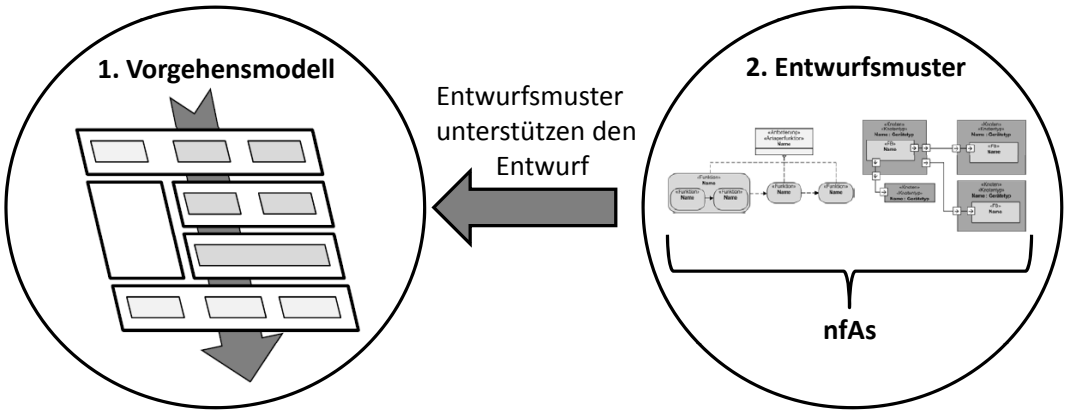

Abbildung 5.1: Übersicht über die Vorgehensweise in dieser Arbeit

Im ersten Schritt zur Systematisierung des Entwurfsprozesses wird das V-Modell so erweitert, dass die in Abschnitt 2.4 genannten Herausforderungen in Bezug auf verteilte Steuerungsarchitekturen Berücksichtigung finden. Somit werden alle Problem- und Aufgabenstellungen des Entwurfs verteilter Automatisierungssysteme in das V-Modell eingebunden, sodass dieses grundsätzlich in abgewandelter Form angewendet werden kann. Die zugehörige Definition des Vorgehensmodells für verteilte Steuerungsarchitekturen sowie dessen Aktivitäten und die Einbindung von nfAs werden in Kapitel 6 erläutert.

Das sich anschließende Kapitel 7 beinhaltet einen Vorschlag für das Entwurfsmusterkonzept, das den Anwendungsentwickler beim Entwurf mittels Entwurfsvorschlägen unterstützt und somit den zweiten Schritt im Rahmen der Entwurfsunterstützung von verteilten Automatisierungssystemen bildet. Im Hinblick auf einen anforderungsgetriebenen Entwurf sollten die nfAs in die Entwurfsmuster eingebunden werden, sodass eine Erfüllung beziehungsweise Nichterfüllung von nfAs bei Anwendung der Entwurfsmuster ersichtlich ist. Anschließend erfolgt in Kapitel 7 eine Beschreibung der Notation, die für die Darstellung der Entwurfsmusterlösung verwendet wird.

Ausgehend vom Entwurfsmusterkonzept erfolgt in Kapitel 8 eine Ableitung möglicher Entwurfsmuster. Hierzu wurde eine Klassifizierung und Sammlung möglicher Automatisierungsaufgaben anhand einer Analyse mehrerer Automatisierungsanlagen – verfahrens- und prozesstechnische Anlagen – vorgenommen. Aus dieser Klassifizierung und Sammlung von Automatisierungsaufgaben können Entwurfsmuster abgeleitet werden. Aufgrund der allgemeingültigen Struktur des Entwurfsmusterkonzepts wird die Bildung neuer Entwurfsmuster ermöglicht.

Abschließend erfolgt in Kapitel 9 eine Anwendung des Konzepts – Vorgehensmodell, nfAs und Entwurfsmuster – am Beispiel einer fertigungstechnischen und einer verfahrenstechnischen Anlage. Hierzu werden die einzelnen Aktivitäten des Vorgehensmodells ausgeführt, die Ausgangsdokumente definiert und die jeweiligen Entwurfsmuster angewendet.

6 Vorgehensmodell für den Entwurf verteilter Automatisierungssysteme

Um das Engineering von verteilten Automatisierungssystemen gezielt zu unterstützen und zu vereinfachen, wurde im Jahre 2010 das Forschungsprojekt »Funktionaler Anwendungsentwurf für verteilte Automatisierungssysteme (FAVA)« initiiert. Ziel des von der »Deutschen Forschungsgemeinschaft (DFG)« geförderten Forschungsvorhabens war es, eine Methodik für den systematischen Entwurf verteilter Automatisierungssysteme zu entwickeln. Durch das Forschungsvorhaben soll der Anwendungsentwickler mithilfe eines systematischen Vorgehensmodells in die Lage versetzt werden, die gewünschte automatisierungstechnische Funktionalität (abgeleitet aus funktionalen Anforderungen) eines Automatisierungssystems zu entwerfen. Des Weiteren wird der Anwendungsentwickler dabei unterstützt, die nfAs zu berücksichtigen. Das in den nachfolgenden Abschnitten dieses Kapitels vorgestellte Vorgehensmodell sowie die Einordnung der nfAs wurden in Zusammenarbeit mit dem *Lehrstuhl für Automatisierung und Informationssysteme* (AIS) der Technischen Universität München und dem *Institut für Automatisierungstechnik* (IFAT) der Otto-von-Guericke-Universität Magdeburg entwickelt [Frank et al. 2011]* [Frank et al. 2012a]* [Frank et al. 2013a]*.

6.1 Beschreibung des Vorgehensmodells

In den folgenden Abschnitten wird das Vorgehensmodell für verteilte Automatisierungssysteme erläutert. Hierzu wird einerseits das Vorgehensmodell allgemein beschrieben und andererseits dessen Aktivitäten und Ausgangsdokumente.

6.1.1 Ebenenweise Vorgehensweise

Durch die zunehmende Komplexität von Automatisierungssystemen und den vermehrten Einsatz immer anspruchsvollerer Technik werden Automatisierungssysteme anfälliger für Störungen und Fehler (beispielsweise Prozessstörungen), die sich räumlich begrenzen beziehungsweise reduzieren lassen, indem die Funktionalität gezielt im System verteilt wird. Die stets wachsende Anzahl von leistungsfähigen Knoten und die damit einhergehenden Anforderungen an die Automatisierungssysteme haben zur Folge, dass es unabdingbar ist, eine verteilte Steuerungsarchitektur aufzubauen.

Da existierende Vorgehensmodelle, wie beispielsweise V-Modell, V-Modell xt oder NA 35, nicht speziell auf verteilte Automatisierungssysteme ausgerichtet sind, können diese insoweit nur unter erschwerten Bedingungen Anwendung finden, da die Aktivitäten und Dokumente nicht auf die Herausforderungen eines verteilten Automatisierungssystems ausgerichtet sind. Deshalb benötigt der systematische und anforderungsgetriebene Entwurf von verteilten Systemen ein Vorgehensmodell, das speziell auf den Anwendungsfall sowie die Domäne – *verteilte Automatisierungssysteme* – ausgerichtet ist.

Das V-Modell sowie das V-Modell xt beschreiben beide eine »top down«-Vorgehensweise für den Entwurf eines Systems. In Bezug auf den Aspekt *verteilte Automatisierungssysteme* bedeutet dies, dass (nicht präzisierte) Funktionalität nachträglich detailliert werden muss, um die Funktionen den Knoten zuzuweisen, auf denen die Funktionen in weiteren Engineering-Schritten implementiert werden.

Das Vorgehensmodell, das speziell an verteilten Automatisierungssystemen ausgerichtet ist (siehe Abbildung 6.1), basiert auf dem in der Industrie sehr verbreiteten generischen Vorgehensmodell – V-Modell (siehe Tabelle 6.1 auf der nächsten Seite) – mit besonderem Augenmerk auf die Engineering-Dokumente, basierend auf dem V-Modell xt. Hierzu wurde der linke Teil des V-Modells (siehe Abbildung 6.1 – links oben) um einen separaten Entwurfsschritt erweitert, der auf die Verteilung der Funktionalität fokussiert. Des Weiteren wurden die einzelnen Aktivitäten speziell an die Herausforderungen verteilter Automatisierungssysteme angepasst. Der Entwurf, basierend auf nachfolgend erläutertem Vorgehensmodell, startet mit einer Anforderungsanalyse, auf Basis derer der Entwurf detailliert wird. Das Ziel des Vorgehensmodells ist das Ausführen des Engineerings und das gezielte Konkretisieren der Verteilung auf unterschiedlichen Abstraktionsebenen (*Ebene 4* bis *Ebene 1*). Diese stufenweise Verfeinerung erlaubt das gezielte Betrachten von nfAs im Entwurfsprozess.

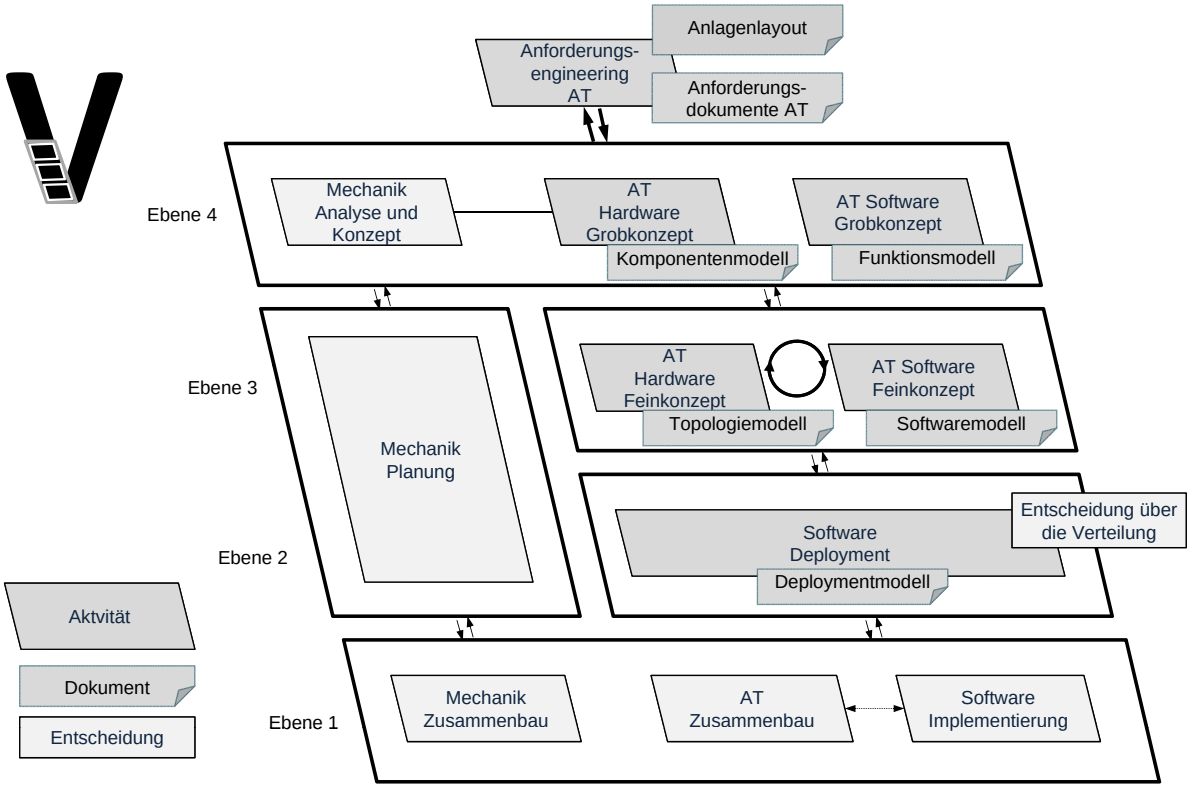

Abbildung 6.1: Vorgehensmodell für verteilte Automatisierungssysteme

Die Abbildung 6.1 fokussiert auf den linken Teil des V-Modells, in dem die Systemanforderungen sukzessive in einen Systementwurf umgesetzt werden. Das Vorgehensmodell segmentiert den Entwurf in drei Säulen – Mechanik, automatisierungstechnische Hardware (AT Hardware) und automatisierungstechnische Software (AT Software) sowie in vier Ebenen – Grobentwurf, Feinentwurf, Deployment (Verteilung) und Implementierung.

Diese Aufteilung in Teilgebiete ist vorteilhaft, weil *Ebene 4* dem funktionalen und technologisch unabhängigen Aspekt des Systems entspricht, *Ebene 3* die logische Struktur des Systems darstellt, *Ebene 2* die technologiespezifische Struktur (Verteilung) des Systems repräsentiert und *Ebene 1* die Implementierung des Systems beinhaltet [vgl. Frank et al. 2012a, S. 297]*. In Tabelle 6.1 ist ein Vergleich der Ebenenstruktur des Vorgehensmodells für verteilte Automatisierungssysteme und des V-Modells dargestellt.

Tabelle 6.1: Vergleich des Vorgehensmodells für verteilte Automatisierungssysteme mit dem V-Modell

	Vorgehensmodell für verteilte Automatisierungssysteme	V-Modell
Ebene	Grobentwurf	Grobentwurf
	Feinentwurf	Feinentwurf
	Deployment	–
	Implementierung	Implementierung

Der parallele Entwurf von Mechanik, AT Hardware und AT Software, die eng definierte Kopplung zwischen diesen Disziplinen, die Definition von Ausgangsdokumenten für jeden Entwurfsschritt sowie die starke Einbindung des Aspekts Verteilung (Deployment) ermöglichen einen Entwurf, der eine Verteilung von Funktionalität unterstützt. Hierzu wurde die Verteilung als einer von mehreren Entwurfsschritten in das Vorgehensmodell integriert.

6.1.2 Aktivitäten und Ausgangsdokumente der einzelnen Ebenen

Das Vorgehensmodell strukturiert den Entwurfsprozess in eine Ebenenstruktur, bestehend aus vier Ebenen beziehungsweise Entwurfsschritten (*Ebene 4* bis *Ebene 1*). Jeder Entwurfsschritt ist charakterisiert durch Aktivitäten, in denen, basierend auf den Anforderungen an das zu entwickelnde System sowie den Ergebnissen in den vorherigen Ebenen, die Lösung der jeweiligen Ebene entworfen wird. Zum Abschluss eines jeden Entwurfsschritts werden die Ergebnisse beziehungsweise Lösungen in Ausgangsdokumenten festgehalten, die den nachfolgenden Ebenen als Eingangsdokument dienen.

Die Aktivitäten und die Ausgangsdokumente des Vorgehensmodells sind an die Ebenen (*Ebene 4* bis *Ebene 1*) und Säulen (Mechanik, AT Hardware und AT Software) angepasst und erlauben eine Iteration zwischen diesen Ebenen. Im Nachfolgenden werden die Aktivitäten sowie die Ausgangsdokumente der einzelnen Ebenen beschrieben. Da die Mechanik üblicherweise nicht vom Planer verändert werden kann, wird diese im Folgenden nicht näher erläutert.

Die Darstellung der Ergebnisse in den jeweiligen Ausgangsdokumenten der Ebenen basiert auf der vom ais entwickelten und in [Frank et al. 2013b] beschriebenen Notation.

Ausgangspunkt

Ausgangspunkt des Vorgehensmodells ist eine Anforderungsanalyse, die aus einem Anforderungsdokument der Automatisierungstechnik resultiert (siehe Abbildung 6.2 auf der nächsten Seite). Diese Anforderungsdokumente, wie beispielsweise R&I-Fließbild, Beschreibung des zu automatisierenden technischen Prozesses oder in der Fertigungstechnik das Anlagenlayout, beinhalten die funktionalen und nicht-funktionalen Anforderungen.

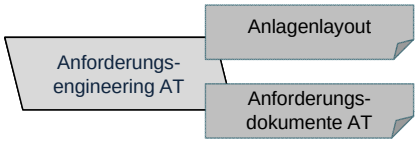

Abbildung 6.2: Aktivitäten und Dokumente des Ausgangspunkts

Ebene 4

Ausgehend von den Anforderungsdokumenten der Automatisierungstechnik, werden in der Aktivität *AT Hardware Grobkonzept* die automatisierungstechnischen Geräte (Sensoren und Aktoren) ermittelt, die zur Erfüllung der funktionalen Anforderung notwendig sind. Die Ergebnisse der Aktivität werden im Ausgangsdokument *Komponentenmodell* dargestellt (siehe Abbildung 6.3).

In Produktionssystemen ist die Struktur des *Komponentenmodells* von der Lage der Sensoren und Aktoren in der Produktionsanlage sowie von den Umgebungsbedingungen, wie beispielsweise Feuchtigkeit und Staub, abhängig. An die Sensoren und Aktoren werden unterschiedliche nfAs, wie beispielsweise Ressourcennutzung, gestellt. Zusätzlich erfolgt auf dieser Ebene in der Aktivität *AT Software Grobkonzept* eine hierarchische Zerlegung von funktionalen Anforderungen (Anlagenfunktionen) bis hin zu automatisierungstechnischen Grundfunktionen (Automatisierungsfunktionen). Somit werden auf dieser Ebene ein oder mehrere Automatisierungsfunktionen ausgewählt, welche die geforderte funktionale Anforderung (Anlagenfunktion) realisieren. Im Ausgangsdokument *Funktionsmodell* werden diese Ergebnisse dargestellt (siehe Abbildung 6.3). Dieser Ansatz ist vorteilhaft, weil die Funktionen unabhängig von nfAs detailliert werden können und die nfAs, wie beispielsweise Zeitanforderungen, zwischen Funktionen sichtbar werden. Diese Anforderungen werden dann in einer der nachfolgenden Ebenen berücksichtigt.

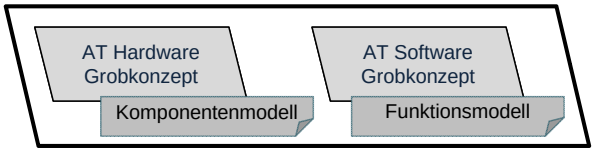

Abbildung 6.3: Aktivitäten und Dokumente auf Ebene 4

Basierend auf der in [Frank et al. 2013b] eingeführten Notation, zeigt Abbildung 6.4 eine mögliche Darstellung der Ergebnisse des Entwurfsschritts in den Ausgangsdokumenten *Komponentenmodell* und *Funktionsmodell*. Abbildung 6.4 zeigt eine allgemeingültige Darstellung ohne Bezug auf ein Anwendungsbeispiel.

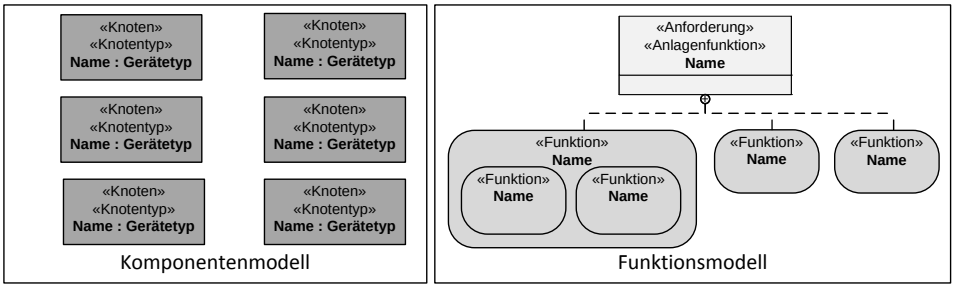

Abbildung 6.4: Darstellung der Ergebnisse auf Ebene 4 (anwendungsneutral)

Ebene 3

Auf *Ebene 3* besteht eine Verbindung zwischen den Aktivitäten *AT Hardware Feinkonzept* und *AT Software Feinkonzept.* In der Aktivität *AT Hardware Feinkonzept* können mithilfe der benötigten Funktionalität der Sensoren und Aktoren die konkreten Geräte ausgewählt werden. Die Ergebnisse der Aktivität werden im Ausgangsdokument *Topologiemodell* dargestellt (siehe Abbildung 6.5). Zusätzlich erfolgt auf dieser Ebene in der Aktivität *AT Software Feinkonzept* die Spezifizierung der Automatisierungsfunktionen durch ein oder mehrere Funktionsblöcke, in die das spätere Sollverhalten, die Definition des internen Verhaltens der Funktionsblöcke sowie die Verbindung von Funktionsblöcken über Schnittstellen (Ports) implementiert werden. Zusätzlich erfolgt auf dieser Ebene die Verknüpfung der Funktionsblöcke mit der benötigten automatisierungstechnischen Hardware, um aufzuzeigen, dass ein Informationsaustausch zwischen Funktionsblock und Hardware erfolgt. Die Ergebnisse der Aktivität werden im Ausgangsdokument *Softwaremodell* dargestellt (siehe Abbildung 6.5). Das *Topologiemodell* sowie das *Softwaremodell* sind nicht unabhängig voneinander, weil auf dieser Ebene zusätzlich die benötigten Funktionsblöcke mit der benötigten Automatisierungshardware verknüpft werden und somit die Interoperabilität spezifiziert wird (siehe Abbildung 6.5).

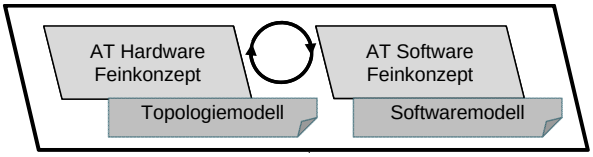

Abbildung 6.5: Aktivitäten und Dokumente auf Ebene 3

Abbildung 6.6 zeigt eine mögliche anwendungsneutrale Darstellung des *Topologiemodells* sowie des *Softwaremodells*, in der die Automatisierungshardware mit den Funktionsblöcken verknüpft wurde.

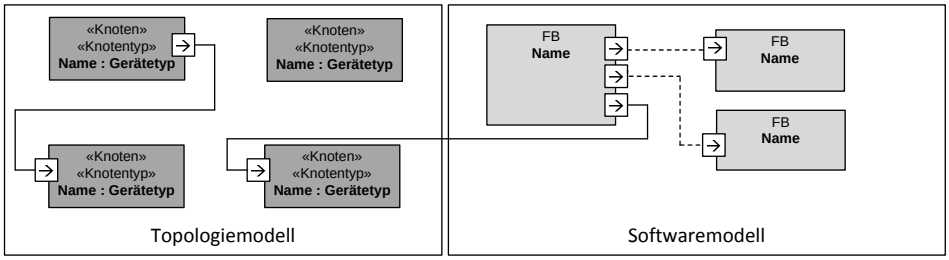

Abbildung 6.6: Darstellung der Ergebnisse auf Ebene 3 (anwendungsneutral)

Ebene 2

Basierend auf den Ergebnissen von *Ebene 3*, wird auf dieser Ebene im Ausgangsdokument *Deploymentmodell* die Verteilung der Funktionsblöcke auf die Steuerungshardware beschrieben (siehe Abbildung 6.7 auf der nächsten Seite). Bei der Entscheidung bezüglich einer möglichen Verteilungsart ist es notwendig, die Ausprägungen der nfAs zu beachten. Hierzu müssen die

Anzahl und die Art der Steuerungshardware sowie das Kommunikationssystem festgelegt werden. Ebenso wird auf dieser Ebene die Ausführungsumgebung jedes einzelnen Funktionsblocks bestimmt. Hierzu müssen verschiedene Verteilungsalternativen evaluiert und die Systemarchitektur und die Kommunikationssysteme gewählt werden, welche die nfAs erfüllen.

Typischerweise gelangt der Anwendungsentwickler aufgrund der Komplexität eines Systems nicht auf Anhieb zu einer guten Lösung, sondern nähert sich dieser in Iterationen, dargestellt im Vorgehensmodell auf *Ebene 3* als Kreis (siehe Abbildung 6.5 auf der vorherigen Seite). Diese Iterationen verursachen Fehler in der Installation und dadurch erhöhte Kosten. In der Praxis führen solche Iterationen zu »trial and error«-Methoden, wenn nicht auf in der Vergangenheit bewährte Lösungen zurückgegriffen wird. Das *Deploymentmodell* wird als Grundlage für die spätere Implementierung auf *Ebene 1* genutzt.

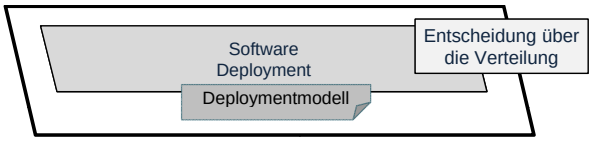

Abbildung 6.7: Aktivitäten und Dokumente auf Ebene 2

Abbildung 6.8 zeigt eine mögliche anwendungsneutrale Darstellung des *Deploymentmodells*, in der die Funktionsblöcke auf die Steuerungshardware verteilt wurden.

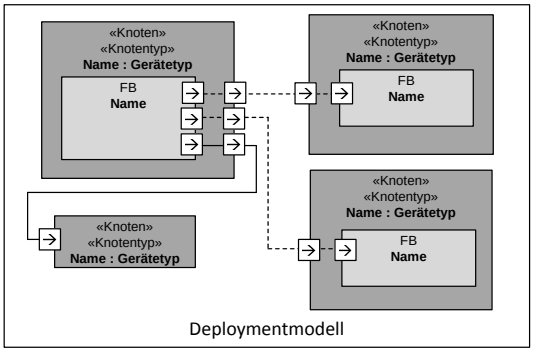

Abbildung 6.8: Darstellung der Ergebnisse auf Ebene 2 (anwendungsneutral)

Ebene 1

Nachdem alle Designentscheidungen getroffen wurden, werden auf *Ebene 1* auf Grundlage des *Deploymentmodells* die Funktionsblöcke in einer der fünf SPS-Programmiersprachen der IEC 61 131-3 [IEC 61 131-3 2003][#] implementiert (siehe Abbildung 6.9). Die Implementierung des Steuerungscodes ist nicht Teil dieser Arbeit.

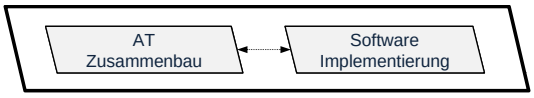

Abbildung 6.9: Aktivitäten auf Ebene 1

Zusammenfassung

Durch die systematische und anforderungsgetriebene Herangehensweise an den Entwurf von verteilten Automatisierungssystemen ist ein gezieltes Konkretisieren der Verteilung möglich. Hierzu ist es notwendig, dass als einer der zentralen Schritte Anzahl und Art der Steuerungshardware ausgewählt werden, in dem die Funktionsblöcke verteilt werden können. Deshalb

wurde der Aspekt Verteilung als ein Entwurfsschritt in das Vorgehensmodell integriert (*Ebene 2*). Jede Ebene des Entwurfsprozesses besteht aus den zuvor beschriebenen Aktivitäten und Ausgangsdokumenten, die den nachfolgenden Ebenen als Grundlage für ihren Entwurfsschritt dienen. Tabelle 6.2 zeigt zusammengefasst die einzelnen Aktivitäten des Vorgehensmodells bezogen auf die Ebenen.

Tabelle 6.2: Aktivitäten des V-MODELLS in Bezug auf seine Ebenen [vgl. FRANK ET AL. 2013a, S. 82]*

Ebene im Vorgehensmodell	Tätigkeit	
	AT Hardware	AT Software
Ebene 4	• Ermittlung der benötigten AT-Gerätetypen	• Hierarchische Zerlegung der funktionalen Anforderungen (Anlagenfunktionen) in Automatisierungsfunktionen
Ebene 3	• Feldgeräte auswählen	• Automatisierungsfunktionen in Funktionsblöcke zerlegen • Internes Verhalten der Funktionsblöcke festlegen • Schnittstellen spezifizieren • Verbindung der Funktionsblöcke über Schnittstellen
	• Funktionsblöcke mit der benötigten Automatisierungshardware verknüpfen	
Ebene 2	• Anzahl Knoten auswählen	
	• Funktionsblöcke auf Knoten verteilen • Knoten-Knoten-Kommunikation festlegen und Schnittstellen spezifizieren	
Ebene 1	• Hardwareaufbau	• Implementierung

6.2 Integration von Anforderungen in das Vorgehensmodell

In den weiteren Abschnitten wird die Einordnung von Anforderungen in das zuvor beschriebene Vorgehensmodell erläutert. Hierzu wird zunächst analysiert, welche nfAs in verteilten Automatisierungssystemen relevant sind. Anschließend werden die funktionalen Anforderungen in Anlagen- und Automatisierungsfunktionen unterteilt, um die unterschiedlichen Detaillierungsgrade der funktionalen Anforderung in das Vorgehensmodell einzuordnen.

6.2.1 Berücksichtigung nicht-funktionaler Anforderungen

Aufgrund der Komplexität und Kopplung zwischen Automatisierungsgeräten, Steuerungssystemen sowie der Funktionalität müssen an verteilte Automatisierungssysteme andere nfAs gestellt werden als beispielsweise an Systeme mit einer zentralen Steuerungsarchitektur. Deshalb muss eine Integration von nfAs in das zuvor beschriebene Vorgehensmodell stattfinden.

Während die funktionalen Anforderungen (= geforderte Anlagenfunktionen) die Funktionalität der Anlage festlegen (zum Beispiel Flüssigkeit mischen oder Flüssigkeit transportieren), definieren nfAs allgemeine Eigenschaften einer Anlage, einer Funktion beziehungsweise eines Betriebsmittels [vgl. ISO/IEC 25 010 2011, S. 10 ff.]#. Im Gegensatz zu den funktionalen Anfor-

derungen, die beschreiben, »was« ein System leisten soll (Funktionalität des Systems), geben die nfAs an, »wie gut« ein System etwas leisten soll (Qualität der Funktion) [V-Modell xtb][@]. Die nfAs können sich auf die Entwurfsumgebung, das Automatisierungssystem sowie auf Elemente des Automatisierungssystems beziehen. Nicht-funktionale Anforderungen sind schwierig spezifizierbar und werden deshalb oftmals vernachlässigt [vgl. Zou & Pavlovski 2006, S. 316]. In [Davis & Leffingwell 1996] ist beschrieben, dass viele Fehler in Softwareprojekten auf unzureichende Anforderungen zurückzuführen sind. Deshalb ist die Berücksichtigung und Einhaltung von nfAs von enormer Wichtigkeit. Die allgemeinen Qualitäts- und Softwaremodelle, wie beispielsweise ISO/IEC 25 010 [ISO/IEC 25 010 2011][#] oder ISO/IEC 9126 [ISO/IEC 9126 2002][#], beschreiben funktionale und nicht-funktionale Anforderungen an Software im Allgemeinen und berücksichtigen nicht spezielle Anforderungen an verteilte Automatisierungssysteme. Da sich bestimmte nfAs in einer verteilten Steuerungsarchitektur verschärfen können beziehungsweise schwieriger zu erfüllen sind, müssen diese explizit beachtet werden, wie beispielsweise das Zeitverhalten.

Im Folgenden werden zuerst die nfAs beschrieben, die einfacher in einer zentralen Steuerungsarchitektur erfüllt werden (nfA Installierbarkeit bis Modifizierbarkeit), und dann die nfAs, die einfacher in einer verteilten Steuerungsarchitektur zu erfüllen sind (nfA Modularität bis Fehlertoleranz). Die nachfolgenden beschriebenen nfAs sind speziell in einer verteilten Steuerungsarchitektur von hoher Wichtigkeit.

Installierbarkeit: Die Installierbarkeit beschreibt den Erfüllungsgrad, mit dem ein System erfolgreich in eine bestimmte Umgebung integriert werden kann [vgl. ISO/IEC 25 010 2011, S. 15][#]. In einem verteilten System ist die Installation durch die vielen Installationsumgebungen schwieriger. Im Vergleich zur zentralen Steuerungsarchitektur ist der Kommunikationsaufwand zwischen den einzelnen Komponenten wesentlich erhöht.

Wiederverwendbarkeit: Die Wiederverwendbarkeit beschreibt das Maß, zu dem ein System mit minimalem Aufwand in mehr als einem System verwendet werden kann [vgl. ISO/IEC 25 010 2011, S. 15][#]. In einem verteilten System ist eine Wiederverwendung durch die vielen Kommunikationsmöglichkeiten und dem daraus resultierenden erhöhten Implementierungsaufwand schwieriger als beispielsweise in einer zentralen Steuerungsarchitektur.

Analysierbarkeit: Die Analysierbarkeit ist das Maß, mit dem die Auswirkungen einer Änderung bewertet sowie Fehler und Mängel innerhalb einer Anlage aufgefunden beziehungsweise zurückverfolgt werden können [vgl. ISO/IEC 25 010 2011, S. 15][#]. In einem verteilten System erhöht sich der Aufwand zur Analyse von Fehlern aufgrund vieler Kommunikationswege sowie mehrerer Steuereinheiten.

Testbarkeit: Die Testbarkeit ist ein Maß dafür, mit welchem Aufwand ein System getestet werden kann, und beinhaltet Prüfkriterien, anhand derer die Testbarkeit bewertet werden kann [vgl. ISO/IEC 25 010 2011, S. 15][#]. In einem verteilten System ist die Testbarkeit aufgrund vieler Schnittstellen sowie Abhängigkeiten zwischen den Steuereinheiten schwieriger.

Interoperabilität: Die Interoperabilität beschreibt den Grad, mit dem Komponenten beziehungsweise Systeme Informationen austauschen und bearbeiten können [vgl. ISO/IEC 25 010 2011, S. 12][#]. Aufgrund des steigenden Kommunikationsaufwands zwischen Geräten ist es in einem verteilten System schwieriger, die Interoperabilität einzuhalten.

Zeitverhalten (Echtzeit): Das Zeitverhalten beschreibt das Maß der Reaktionszeiten und der Verarbeitungszeiten sowie die Durchsatzrate eines Systems während des Betriebs [vgl. ISO/IEC 25010 2011, S. 11][#]. In einem verteilten System ist es schwieriger, das Zeitverhalten einzuhalten, da die Reaktionszeiten aufgrund des erhöhten Kommunikationsaufwands steigen. Beim Entwurf von verteilten Systemen ist es wichtig, dass die Echtzeitbedingungen eingehalten werden. Deshalb kann diese nfA in einem zentralen Steuerungssystem leichter erfüllt werden.

Modifizierbarkeit: Die Modifizierbarkeit ist ein Maß für das effiziente und wirksame Modifizieren eines Systems ohne die Einführung von Fehlern oder die Verschlechterung seiner Performance [vgl. ISO/IEC 25010 2011, S. 15][#]. In einem verteilten System ist die Modifizierbarkeit schwieriger einzuhalten, da Änderungen am System größere Auswirkungen auf andere Systembestandteile haben können, als in einer zentralen Steuerungsarchitektur.

Modularität: Die Modularität bezeichnet den Erfüllungsgrad, zu dem ein System aus diskreten Komponenten zusammengesetzt wird, sodass minimale Abhängigkeiten zwischen den Komponenten bestehen und der Wechsel von Komponenten minimale Auswirkungen hat [vgl. ISO/IEC 25010 2011, S. 14][#]. In verteilten Systemen ist die Erfüllung dieser nfA einfacher zu realisieren, weil durch die verteilte Steuerungsarchitektur ein Ansatz von Modularität vorgegeben ist. In zentralen Steuerungsarchitekturen kann eine Modularität nur durch eine geeignete Implementierung der Software erreicht werden.

Ressourcennutzung: Die Ressourcennutzung beschreibt Quantität und Art der im System erforderlichen Ressourcen, die benötigt werden, um die Funktionalität des Systems zu erfüllen [vgl. ISO/IEC 25010 2011, S. 11][#]. In einem verteilten System ist eine Erfüllung dieser nfA einfacher zu realisieren, da im Gegensatz zu einer zentralen Steuerungsarchitektur durch die Einbindung mehrerer Steuereinheiten der Ressourcenknappheit entgegengewirkt wird.

Fehlertoleranz: Die Fehlertoleranz beschreibt ein Maß, zu dem ein System trotz vorhandener Hardware- und Softwarefehler entsprechend den Bestimmungen funktioniert [vgl. ISO/IEC 25010 2011, S. 13][#]. Fehlertoleranz-Mechanismen, wie beispielsweise Redundanz der Steuerung, können aufgrund ihrer verteilten Struktur in verteilten Systemen leichter realisiert werden. Wenn ein Knoten ausfällt, kann ein anderer Knoten dessen Aufgaben übernehmen. Tritt in einem verteilten System ein Fehler auf, hat dieser nur Auswirkungen auf das Subsystem, und im Gegensatz zum zentralen System ist nicht das gesamte System betroffen.

Die nfAs, die an das verteilte Automatisierungssystem gestellt werden, müssen in unterschiedlichen Schritten des Entwurfsprozesses beachtet und erfüllt werden. Sie können nicht schon bei Projektstart erfüllt werden. Zum Beispiel kann die Ressourcennutzung erst berücksichtigt werden, wenn Anzahl und Art der Knoten feststehen sowie die Softwareverteilung, was in einem späteren Entwurfsschritt geschieht. Bei dem in Abschnitt 6.1 vorgestellten Vorgehensmodell müssen auf jeder der vier Ebenen spezielle für verteilte Systeme relevante nfAs berücksichtigt werden. Deshalb werden die zuvor für verteilte Systeme herausgearbeiteten nfAs den Ebenen zugeordnet. Dadurch wird sichergestellt, dass die nfAs an die Tätigkeiten der Ebenen angepasst sind und erst dort berücksichtigt werden, wo genug Informationen vorliegen, um die nfAs durch geeignete Entwurfsentscheidungen zu erfüllen.

Die Zuordnung der nfAs hat zum Ziel, dass diese erst auf bestimmten Ebenen formuliert, aufgestellt und im Vorgehensmodell auf gleicher oder darunterliegender Ebene als Vorgabe genutzt und erfüllt werden. Diese Zuordnung ist notwendig, weil zu Beginn eines Entwurfsprozesses noch

nicht alle Anforderungen feststehen, sondern manche sich erst während des Engineerings ergeben. Des Weiteren ist es durch die Zuordnung möglich, die nfAs mit den Zwischenergebnissen des Engineerings abzugleichen und dort einzubringen, wo ausreichend Informationen vorliegen. Dadurch wird der Entwurfsprozess von verteilten Automatisierungssystemen vereinfacht.

Einordnung der nfAs in Ebene 4

Die Aktivitäten auf *Ebene 4* (siehe Abschnitt 6.1.2) erlauben eine Einbindung und Berücksichtigung verschiedener nfAs. Aufgrund des hohen Implementierungsaufwands verteilter Automatisierungssysteme sollte die Erarbeitung der Ergebnisse des *AT Hardware Grobkonzepts* sowie des *AT Software Grobkonzepts* und der darunter liegenden Ebenen berücksichtigen, dass bestimmte Systemteile in unterschiedlichen Projekten Anwendung finden können (nfA *Wiederverwendbarkeit*). Dadurch wird in nachfolgenden Systemteilen beziehungsweise Projekten der Aufwand für den Anwendungsentwickler reduziert. Eine weitere nfA, die in die Aktivitäten der *Ebene 4* und der darunterliegenden Ebenen eingeordnet werden kann, ist die *Testbarkeit* von Systemen beziehungsweise Teilsystemen. Aufgrund der hohen Anzahl von Software, Hardware und Schnittstellen muss ein kontinuierliches Testen von Teilsystemen beziehungsweise Systemen im Fokus des Anwendungsentwicklers liegen.

Einordnung der nfAs in Ebene 3

Aufgrund der Aktivitäten auf *Ebene 3* (siehe Abschnitt 6.1.2) können verschiedene nfAs in diese Ebene eingeordnet und berücksichtigt werden. Die Verbindung von Funktionsblöcken und Hardware hat zur Folge, dass die *Interoperabilität* spezifiziert wird, um Informationen austauschen zu können. Dies bedeutet, dass zum Informationsaustausch zwischen Funktionsblöcken und Hardware bestimmte Interoperabilitätskriterien erfüllt werden müssen. Eine weitere nfA, die in die Aktivitäten der *Ebene 3* und der darunterliegenden Ebenen eingeordnet werden kann, ist der modulare Aufbau der Systemarchitektur (nfA *Modularität*). Die Verbindung zwischen Funktionsblöcken und Hardware sollte so aufgebaut werden, dass minimale Abhängigkeiten dazwischen entstehen. Eine spätere verteilte Steuerungsarchitektur gibt zumeist einen Ansatz von Modularität vor. Die *Analysierbarkeit* ist eine weitere nfA, die auf *Ebene 3* eingeordnet werden kann. Aufgrund der vielen Verbindungen zwischen Funktionsblöcken und Hardware steigt die Fehleranfälligkeit. Diese Verbindungen müssen im Entwurf so vorgenommen werden, dass ein Auffinden und Zurückverfolgen von Fehlern ermöglicht wird. Die Analysierbarkeit erlaubt das Auffinden und Zurückverfolgen dieser Fehler. Die letzte nfA, die auf *Ebene 3* eingeordnet wird, ist die *Fehlertoleranz.* Ziel dieser nfA ist, dass bei einem partiellen Ausfall von Hardware und Software nicht ein Gesamtausfall des Systems erfolgt.

Einordnung der nfAs in Ebene 2

Gemäß den in Abschnitt 6.1.2 beschriebenen Aktivitäten der Verteilung müssen auf dieser Ebene bestimmte nfAs berücksichtigt werden. Eine dieser nfAs ist die *Ressourcennutzung.* Diese nfA kann auf *Ebene 2* eingeordnet werden, da sie erst ermittelt werden kann, wenn die Software auf die Hardware verteilt wird. Die nfA *Zeitverhalten* kann auch erst auf *Ebene 2* berücksichtigt werden, da hier die Kommunikationssysteme sowie die Steuerungsarchitektur ausgewählt werden, die beide einen Einfluss auf das Zeitverhalten haben. Die letzte nfA auf *Ebene 2* ist die *Installierbarkeit.* Durch die Verteilung der Software auf die Hardware muss diese in unterschiedliche Ausführungsumgebungen integriert werden.

Einordnung der nfAs in Ebene 1

Die Aktivität auf *Ebene 1* beinhaltet die Implementierung der in den vorherigen Ebenen spezifizierten Software. Diese Implementierung muss die nfA *Modifizierbarkeit* berücksichtigen, sodass diese schnell und einfach an sich ändernde Anforderungen angepasst werden kann.

Zusammenfassung

Tabelle 6.3 zeigt die Zuordnung der zuvor beschriebenen nfAs zu den vier Ebenen des Vorgehensmodells. Diese Tabelle zeigt auf, dass die nfAs auf bestimmten Ebenen formuliert werden und der gleichen oder darunterliegenden Ebene als Vorgabe dienen.

Tabelle 6.3: Zuordnung der nfAs zu den vier Ebenen des Vorgehensmodells

Ebene im Vorgehensmodell	zu berücksichtigende nfA
Ebene 4	• Wiederverwendbarkeit • Testbarkeit
Ebene 3	• Interoperabilität • Modularität • Analysierbarkeit • Fehlertoleranz
Ebene 2	• Ressourcennutzung • Zeitverhalten (Echtzeit) • Installierbarkeit
Ebene 1	• Modifizierbarkeit

Oftmals sind die zuvor beschriebenen nfAs zu abstrakt beschrieben, um später Systemeigenschaften daraus ableiten zu können. Deshalb stellt das IFAT in [Frank et al. 2013a]* ein Merkmalsmodell vor, in dem die Möglichkeit beschrieben wird, nfAs weiter zu untergliedern. Das IFAT beschreibt, dass jede Anforderung, jedes Lösungselement sowie das Entwurfsergebnis mit Merkmalen der nfAs beschrieben werden kann. Hierzu findet eine Unterteilung der Merkmale in Anforderungs- sowie Lösungsmerkmale statt. Mithilfe dieser Merkmale kann auf jedem Entwurfsschritt des Vorgehensmodells eine Überprüfung der Anforderungen an das System stattfinden und der Systementwurf nachgebessert werden.

6.2.2 Beschreibung nicht-funktionaler Anforderungen

Dieses Kapitel beschreibt die Relevanz von nfAs bei der Entwurfsunterstützung. Hierzu wird zunächst erläutert, wie die nfAs vorliegen sollten, um ihre Integration in das zuvor beschriebene Vorgehensmodell für verteilte Automatisierungssysteme zu ermöglichen. Anschließend wird ein Konzept für die Überprüfung der nfAs vorgestellt. Der nachfolgend beschriebene Ansatz zur Evaluation von nfAs basiert auf dem vom IFAT entwickelten und in [Frank et al. 2013a]* erläuterten Merkmalskonzept.

Beschreibung der zu erfüllenden nicht-funktionalen Anforderung

Das Engineering automatisierter Anlagen besteht aus einer Vielzahl von Phasen und erstreckt sich von der Anforderungserhebung bis zur Inbetriebnahme [vgl. Fay 2009, S. 44–48]. Der Entwurf verteilter Automatisierungssysteme umfasst üblicherweise alle Tätigkeiten, die benötigt werden, um die funktionalen Anforderungen zu analysieren und in ein mit der IEC 61 131-3

konformes Steuerungsprogramm umzusetzen. In der Anforderungsanalyse werden die funktionalen Anforderungen und alle dazugehörenden – nicht-funktionalen – Anforderungen, die an das Automatisierungssystem gestellt werden, grundsätzlich in textueller Form in einem Anforderungsdokument der Automatisierungstechnik festgehalten, wie beispielsweise Lasten- oder Pflichtenheft [VDI 3694 2008]#, R&I-Fließbild, Beschreibung des zu automatisierenden technischen Prozesses oder des Anlagenlayouts. Durch die textuellen Formulierungen der Anforderungen in Form von Lasten- und Pflichtenheften kann nicht sichergestellt werden, dass alle Beteiligten der unterschiedlichen Phasen oder Gewerke ein einheitliches Verständnis für die Anforderungen haben. Aufgrund ihres informellen Charakters können diese textuell beschriebenen Anforderungen nur schwierig auf Vollständigkeit überprüft beziehungsweise getestet werden. Sie haben sich dennoch in der Praxis dank der einfachen textuellen Beschreibung durchgesetzt. Deshalb setzt das in Abschnitt 6.1 beschriebene Vorgehensmodell kein neu entwickeltes Beschreibungsmittel für funktionale und nicht-funktionale Anforderungen voraus, sondern unterstützt den Entwurf mit bestehenden Beschreibungsmitteln.

Der Entwurf von Automatisierungssystemen beginnt grundsätzlich damit, dass die im Anforderungsdokument dokumentierten funktionalen Anforderungen analysiert sowie interpretiert werden und in einer der nachfolgenden Phasen beziehungsweise Arbeitsschritte gelöst und implementiert werden. Die Herausforderung der Lösungsfindung steigt in verteilten Automatisierungssystemen durch die zunehmende Komplexität und Funktionalität der Anlage sowie der damit einhergehenden verteilten Steuerintelligenz. Deshalb sind die Anforderungen – funktionale und nicht-funktionale Anforderungen – an ein Automatisierungssystem der Ausgangspunkt für die nachfolgenden Phasen und bilden die Grundlage für eine spätere Überprüfung der Automatisierungslösung.

Merkmale der nicht-funktionalen Anforderung

Zumeist sind die textuell beschriebenen Anforderungen zu abstrakt gehalten, um Systemeigenschaften daraus abzuleiten und die nfAs in späteren Entwurfsphasen zu überprüfen. Hierzu müssen nfAs quantifizierbar sein und weiter untergliedert werden. Das Merkmalskonzept [Frank et al. 2013a]* beschreibt nfAs durch Anforderungs- und Lösungsmerkmale.

Merkmale sind spezielle Eigenschaften eines Systems beziehungsweise eines Systemelements und werden genutzt, um Merkmalsträger, wie beispielsweise Automatisierungsfunktion, Knoten, Kommunikationssystem und Gerät, zu charakterisieren [vgl. Frank et al. 2013a, S. 83]*. Diese Merkmale, die in ihrer Bedeutung klar definiert sind, werden als Basis für Entwurfsentscheidungen genutzt, um eine Überprüfung der nfAs zu ermöglichen [vgl. Frank et al. 2013a, S. 83]*.

Um die Erfüllung einer nfA zu überprüfen, wurden Anforderungsmerkmale, wie beispielsweise Ausführungszeit eines Funktionsblocks oder benötigter Speicher, und Lösungsmerkmale, wie zum Beispiel Klemme-Klemme-Reaktionszeit oder Speicherkompatibilität, definiert [vgl. Frank et al. 2013a, S. 83]*. Diese Anforderungs- und Lösungsmerkmale können den speziell in verteilten Systemen relevanten nfAs (siehe Abschnitt 6.2.1) zugeordnet werden. Durch diese Zuordnung und Einordnung der Merkmale in die Ebenen des Vorgehensmodells können diese genutzt werden, um die Anforderungen an das zu entwerfende System zu evaluieren und zu überprüfen. Der Systementwurf kann nachgebessert werden, wenn die Anforderungen nicht erfüllt werden [vgl. Frank et al. 2013a, S. 83]*. Abbildung 6.10 auf der nächsten Seite zeigt die Verwendung von Merkmalen im Entwurfsprozess.

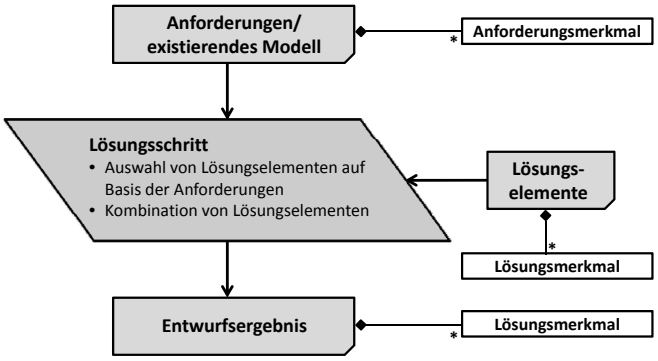

Abbildung 6.10: Verwendung von Merkmalen im Entwurfsprozess [vgl. Frank et al. 2013a, S. 83]*

Das in Abschnitt 6.1 vorgestellte Vorgehensmodell hat den anforderungsgetriebenen Entwurf von verteilten Automatisierungssystemen zum Ziel. Deshalb muss die nachfolgend vorgestellte Entwurfsunterstützung, die auf die Funktionalität eines Systems sowie deren Verteilung abzielt, die relevanten nfAs berücksichtigen, um zu ermitteln, ob die angewandte Verteilung die nfAs erfüllt. Hierzu wird das zuvor vorgestellte Merkmalskonzept aufgegriffen.

6.2.3 Berücksichtigung funktionaler Anforderungen

Beim Entwurf verteilter Systeme müssen nicht nur nfAs, sondern auch funktionale Anforderungen berücksichtigt werden. Die funktionalen Anforderungen (beispielsweise Flüssigkeit mischen, Flüssigkeit transportieren oder Druck protokollieren) legen die Funktionalität einer Anlage fest und sind Grundbestandteil eines jeden Systems beziehungsweise einer jeden Software. Deshalb muss eine Integration von funktionalen Anforderungen in das zuvor beschriebene Vorgehensmodell erfolgen. In unterschiedlichen Stufen des Entwurfsprozesses sind verschiedene Detaillierungsgrade der funktionalen Anforderung relevant. Deshalb wird im Folgenden zwischen Anlagen- und Automatisierungsfunktion unterschieden.

Eine geforderte Anlagenfunktion beschreibt eine stoffliche oder mechanische Verarbeitung – inklusive Informationsverarbeitung –, die zumeist vom Prozessingenieur gefordert wird und in der festlegt ist, was ein System tun soll. Eine Anlagenfunktion, wie beispielsweise Transportieren, Lagerung, Sortierung und Mischen, wird in späteren Engineeringphasen durch eine oder mehrere Automatisierungsfunktionen realisiert.

Eine Automatisierungsfunktion verarbeitet Informationen (beispielsweise Pumpensteuerung, Ventilregelung, Förderbandüberwachung, Füllstandsanzeige) und kann durch beliebig viele andere Automatisierungsfunktionen konkretisiert werden.

Basierend auf dem in Abschnitt 6.1 vorgestellten Vorgehensmodell, spielt die Positionierung der Anlagen- und Automatisierungsfunktionen und somit der Detaillierungsgrad der funktionalen Anforderung eine wichtige Rolle. In der Anforderungsanalyse sind lediglich die Anlagenfunktionen relevant. Auf der *Ebene 4* sind Anlagen- und Automatisierungsfunktionen bedeutsam. Die *Ebene 4* beschreibt den Übergang von Anlagen- zu Automatisierungsfunktionen. Je weiter der Entwurf detailliert wird (ab *Ebene 3*), desto wichtiger sind die Automatisierungsfunktionen, welche die Anlagenfunktionen erfüllen. Diese spielen somit ab *Ebene 3* keine Rolle mehr. Die Automatisierungsfunktionen beschreiben die Funktionalität der Anlage, die auf *Ebene 2* in Form von Funktionsblöcken auf unterschiedliche Knoten verteilt wird.

7 Konzept der Entwurfsmuster für verteilte Automatisierungssysteme

Nachdem in Kapitel 6 sowohl das Vorgehensmodell für verteilte Automatisierungssysteme vorgestellt als auch die Einordnung von nfAs in dieses aufgezeigt wurde, wird im Folgenden eine Entwurfsunterstützung erläutert, die in das Vorgehensmodell eingebunden wird. Auf Basis der Ebenen des Vorgehensmodells wurde ein Musterkonzept entwickelt, das den Anwendungsentwickler beim Entwurf mittels Entwurfsvorschlägen unterstützt und im Weiteren erläutert wird. Ziel dieses Kapitels ist die Darstellung aller für die Entwurfsmuster notwendigen Elemente.

Die Notation, die in den Lösungen der Entwurfsmuster verwendet wird, wurde vom AIS entwickelt und ist in [Frank et al. 2013b] und [Frank 2014] beschrieben. Das Konzept der Merkmale, das zur weiteren Untergliederung der nfAs und zur Ermittlung des Erfüllungsgrads der nfAs notwendig ist, wurde vom IFAT entwickelt und ist in [Frank et al. 2013a]* und [Hadlich 2015] beschrieben.

7.1 Kategorisierung von Entwurfsmustern

In diesem Abschnitt wird eine mögliche Entwurfsunterstützung durch das Konzept der Wiederverwendung beschrieben. Zunächst wird die Entwurfsunterstützung in zwei Kategorien unterteilt, um die Problemstellungen im Entwurf verteilter Automatisierungssysteme zu lösen. Zum einen soll der Anwendungsentwickler bei der Auswahl geeigneter Automatisierungsfunktionen unterstützt werden, zum anderen soll er durch mehrere Vorschläge von Verteilungsalternativen, welche die Erfüllung der nfAs erleichtern, Hilfestellung erfahren. Im Weiteren erfolgt die Einordnung der Entwurfsunterstützung in die Ebenen des Vorgehensmodells. Anschließend werden die beiden Entwurfsmustervorlagen für diese Entwurfsunterstützung dargestellt.

7.1.1 Beschreibung der Kategorien

Die Unterstützung des Anwendungsentwicklers beim Entwurf ist ein wichtiger Schritt bei der Entwicklung verteilter Automatisierungssysteme. Eine detaillierte Analyse der im Vorgehensmodell projektierten Entwurfsschritte und ein exemplarisches Durchlaufen dieses Vorgehensmodells zeigen auf, dass der Anwendungsentwickler auf den unterschiedlichen Ebenen komplexe (Auswahl-)Entscheidungen treffen muss beziehungsweise Entwurfsprobleme zu lösen hat.

Funktionsmuster

Eine dieser Problemstellungen ist das Analysieren von typischen Anlagenfunktionen (funktionalen Anforderungen), die zur Erfüllung der gewünschten Funktionalität der Anlage notwendig sind. Anhand dieser Anlagenfunktionen muss der Anwendungsentwickler die Funktionalität in Form von Automatisierungsfunktionen näher spezifizieren. Die Analyse dieser Problemstellung zeigt auf, dass sie mithilfe von Entwurfsmustern behoben werden kann. Diese sogenannten *Funktionsmuster* beinhalten solche Vorschläge, sodass dem Anwendungsentwickler – auf Basis der zu erfüllenden funktionalen Anforderung – eine Auswahl möglicher Entwurfsmuster zur Verfügung steht. Diese Entwurfsmuster beinhalten unterschiedliche Automatisierungsfunktionen und

können bei Bedarf durch Hinzufügen oder Löschen von Automatisierungsfunktionen angepasst werden. Auf Basis der funktionalen Anforderungen, die in den zuvor erstellten Anforderungsdokumenten enthalten sind, erhält der Anwendungsentwickler eine mögliche Beschreibung der Lösung, in der die Automatisierungsfunktionen Bestandteil sind. Die Abbildung 7.1 zeigt die zuvor beschriebenen Bestandteile eines Funktionsmusters.

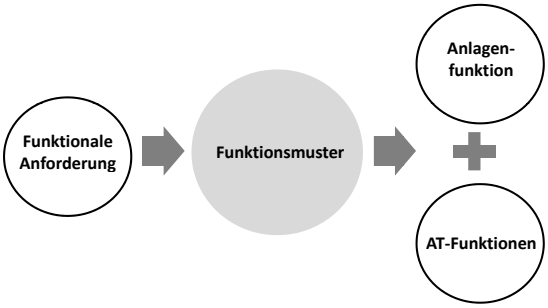

Abbildung 7.1: Aspekte des Funktionsmusters

Verteilungsmuster

Eine weitere Problemstellung im Engineering von verteilten Systemen ist die optimale Verteilung der Funktionalität auf die Knoten des Automatisierungssystems. Hierbei ist es schwierig, eine optimale Verteilung zur Erfüllung von nfAs ohne zusätzliche Hilfestellung zu gewährleisten. Auf Basis dieser Problemstellung wurde ebenfalls eine musterbasierte Entwurfsunterstützung entwickelt, die aufzeigt, welche Auswirkungen die Verteilung der Automatisierungsfunktionen auf die nfAs hat. Die sogenannten *Verteilungsmuster* zeigen mögliche Verteilungsalternativen der Automatisierungsfunktionen. Basierend auf den Automatisierungsfunktionen und den nfAs, erhält der Anwendungsentwickler eine Auswahl möglicher Entwurfsmuster. Auf Grundlage der Anforderungsmerkmale werden die Lösungsmerkmale der nfAs berechnet, sodass für den Anwendungsentwickler ersichtlich ist, ob das ausgewählte Muster die relevanten nfAs erfüllt. Um mithilfe von *Verteilungsmustern* den Entwurf von verteilten Systemen zu unterstützen, müssen in diesen Mustern verschiedene Aspekte berücksichtigt werden (siehe Abbildung 7.2). Basierend auf den Automatisierungsfunktionen und den nfAs, erhält der Anwendungsentwickler eine mögliche Beschreibung der Lösung, die eine Verteilungsalternative beinhaltet.

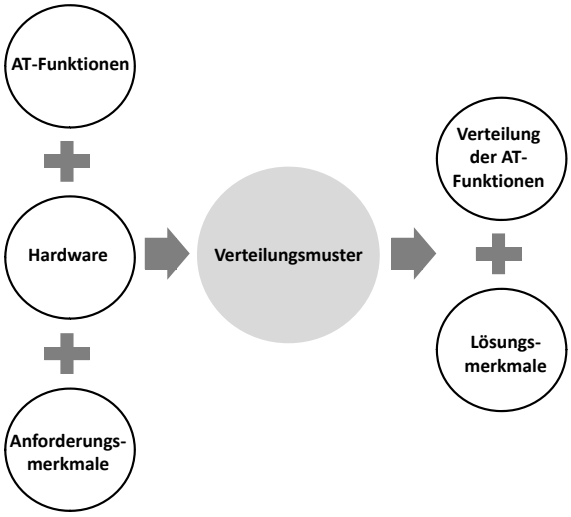

Abbildung 7.2: Aspekte des Verteilungsmusters

Im Verteilungsmuster können zwei verschiedene Verteilungsmethoden für die Automatisierungsfunktionen einer Anlagenfunktion Anwendung finden. In der ersten Verteilungsmethode (siehe Abbildung 7.3 – links) findet eine vertikale Verteilung der Automatisierungsfunktionen statt. Diese bezieht sich auf alle Automatisierungsfunktionen *einer* Anlagenfunktion und beschreibt, wie diese Automatisierungsfunktionen im System verteilt werden. In der zweiten Verteilungsmethode (siehe Abbildung 7.3 – rechts) findet eine horizontale Verteilung der Automatisierungsfunktionen statt. Diese bezieht sich auf alle Automatisierungsfunktionen von *mehreren gleichen* Anlagenfunktionen. Oftmals ist bei der Verteilung der Automatisierungsfunktionen zu beachten, dass beide zuvor beschriebenen Verteilungsmethoden gleichzeitig Anwendung finden.

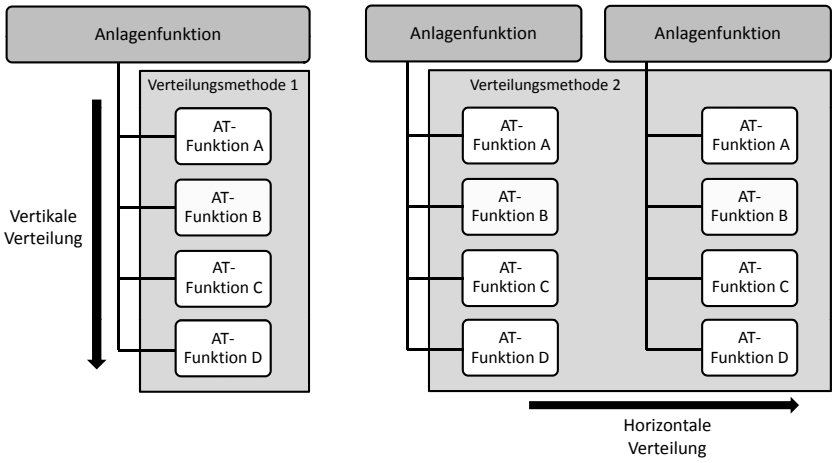

Abbildung 7.3: Verteilungsmethoden

Zusätzlich zu der zuvor beschriebenen vertikalen und horizontalen Verteilung von Automatisierungsfunktionen können diese auf drei verschiedene Möglichkeiten in einem System verteilt werden (siehe Abbildung 7.4). In der ersten Variante ist die jeweilige Automatisierungsfunktion direkt auf dem Knoten II – I/O-Geräte mit Rechenressourcen – und somit direkt am Ort des Entstehens der Funktion implementiert. Die zweite Möglichkeit implementiert die Automatisierungsfunktionen separat auf unterschiedlichen Knoten I – SPS –, während in der dritten Variante die Funktionalität von nur einer Automatisierungsfunktion zentral implementiert und ausgeführt wird.

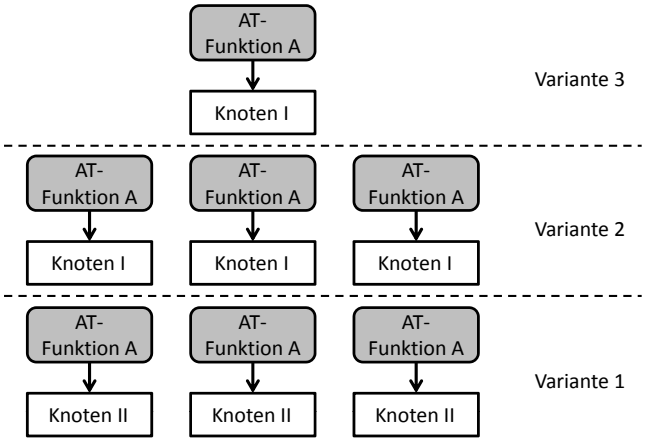

Abbildung 7.4: Verteilungsvarianten

Die zuvor beschriebenen Musterkategorien unterstützen den Anwendungsentwickler also dadurch, dass bewährte Automatisierungslösungen für die geforderten funktionalen Anforderungen angeboten werden (*Funktionsmuster*). Des Weiteren wird der Anwendungsentwickler bei der Verteilung der Funktionalität auf die Automatisierungsgeräte – wie beispielsweise Sensoren, Aktoren und Steuerungen – durch *Verteilungsmuster* unterstützt.

7.1.2 Einordnung der Entwurfsmuster in die Ebenen des Vorgehensmodells

Basierend auf dem in Abschnitt 6.1 eingeführten Vorgehensmodell, sind die Inhalte der Muster sowie deren Anwendung durch den Anwendungsentwickler von der Ebene des Vorgehensmodells abhängig.

Auf *Ebene 4* steht der Anwendungsentwickler in der Aktivität *AT Software Grobkonzept* vor der Problemstellung, dass dieser zwar die Grundfunktionalität (Anlagenfunktion) des Systems kennt (zum Beispiel Transportieren oder Messwert protokollieren), aber die automatisierungstechnischen Grundfunktionen (beispielsweise Weg suchen oder Messwert filtern), die zur Realisierung dieser Funktionalität notwendig sind, erst noch ermitteln muss. Deshalb können die *Funktionsmuster,* die Vorschläge für Automatisierungsfunktionen beinhalten, in *Ebene 4* eingeordnet werden. Auf *Ebene 4* finden sich die *Funktionsmuster* in einer Darstellungsform, die einer »Bibliothek« ähnelt. In ihr werden die *Funktionsmuster* nach funktionalen Anforderungen (Mustertyp) klassifiziert. Für jeden Mustertyp gibt es somit eine Sammlung von *Funktionsmustern.*

Auf *Ebene 2* steht der Anwendungsentwickler in der Aktivität *Software Deployment* vor der Problemstellung, die zuvor ermittelten automatisierungstechnischen Grundfunktionen – Automatisierungsfunktionen – auf die automatisierungstechnische Hardware – Knoten – verteilen zu müssen. Die Automatisierungsfunktionen sollen nicht »beliebig« im System verteilt werden, sondern so, dass die relevanten nfAs erfüllt werden. Deshalb können die *Verteilungsmuster*, die solche Verteilungsvorschläge beinhalten, in *Ebene 2* eingeordnet werden. Aufgrund der zugrunde liegenden Anforderungsmerkmale können bei Anwendung der *Verteilungsmuster* die Lösungsmerkmale der nfAs berechnet werden. Auf *Ebene 2* sind die *Verteilungsmuster* in einer Bibliothek nach funktionalen Anforderungen (Mustertyp) untergliedert. Basierend auf diesen, den Automatisierungsfunktionen und den nfAs, werden dem Anwendungsentwickler unterschiedliche Verteilungsalternativen in Form von *Verteilungsmustern* vorgeschlagen, unter denen er auswählen kann.

7.1.3 Aufbau der Entwurfsmustervorlage

Die verschiedenen Problemstellungen, die der Anwendungsentwickler im Entwurfsprozess lösen muss, haben zur Folge, dass an die Entwurfsmuster unterschiedliche Anforderungen gestellt werden müssen. Deshalb sind die Kategorien der Entwurfsmuster, die den Anwendungsentwickler bei der Lösungsfindung unterstützen sollen, im Inhalt sowie im Problem- und Lösungskontext unterschiedlich. Damit haben die Entwurfsmustervorlagen für die *Funktionsmuster* und die *Verteilungsmuster* einen unterschiedlichen Aufbau.

Funktionsmustervorlage

Der Aufbau der *Funktionsmustervorlage*, basierend auf den Entwurfsmustern der Softwaretechnik, wird im Nachfolgenden vorgestellt.

Musterkategorie: Die *Musterkategorie* vermittelt knapp und präzise die Art des Entwurfsmusters und ist für das schnelle Identifizieren und Einordnen des Problemlösungskontexts notwendig. In den Funktionsmustern hat die *Musterkategorie* immer den Wert »Funktionsmuster«, sodass ersichtlich ist, dass dieses Entwurfsmuster zur Lösung der Problemstellungen auf *Ebene 4* eingeordnet werden kann.

Mustertyp: Der *Mustertyp* beschreibt knapp und präzise für welche Art von Anlagenfunktion das Entwurfsmuster hilfreich ist. Durch den *Mustertyp* ist für den Anwendungsentwickler schnell ersichtlich, welche Entwurfsmuster für diesen Typ von Anlagenfunktion hilfreich sind.

Mustername: Der *Mustername* vermittelt knapp und präzise den wesentlichen Inhalt des Entwurfsmusters. Wie Gamma in den Entwurfsmustern der Softwaretechnik erläutert, beschreibt der *Mustername* das Entwurfsproblem und seine Lösungen mit wenigen Worten [vgl. Gamma et al. 2004, S. 3]. Die Benennung von Mustern mithilfe eines *Musternamens* ist sehr wichtig, weil dieser die Anlagenfunktion beschreibt, für die dieses Entwurfsmuster geeignet ist.

Zweck: Der *Zweck* beschreibt das Grundprinzip sowie Sinn und Ziel des Entwurfsmusters und wird benötigt, um zu erläutern, welche spezifischen Probleme und Fragestellungen das Entwurfsmuster behandelt.

Kontext: Der *Kontext* beschreibt den Anwendungs- und Nutzungskontext, um zu vermitteln, in welchen Situationen das Entwurfsmuster angewendet werden kann. Des Weiteren beschreibt der *Kontext* die Problemsituation, in der das Entwurfsmuster helfen kann.

Lösung: Die grafische Darstellung der *Lösung* ermöglicht eine einfache Repräsentation aller Funktionen, die in diesem Muster relevant sind sowie ihre Beziehungen und Interaktionen. Basierend auf Gamma, beschreibt die *Lösung* keinen bestimmten Entwurf und auch keine konkrete Implementierung, sondern lediglich eine Schablone, die in vielen verschiedenen Situationen angewendet werden kann [vgl. Gamma et al. 2004, S. 4]. Die *Lösung* ermöglicht eine abstrakte Beschreibung des Entwurfsproblems – Anlagenfunktion – und zeigt auf, wie eine allgemeine Anordnung von Elementen – Automatisierungsfunktionen – dieses Problem löst [vgl. Gamma et al. 2004, S. 4].

Teilnehmer: Dieser Abschnitt beschreibt textuell die Lösung des Entwurfsmusters, um dem Anwendungsentwickler ein einheitliches Verständnis der Funktionen zu vermitteln.

Verteilungsmustervorlage

Die Vorlage der *Verteilungsmuster* hat aufgrund der zu lösenden Problemstellung auf *Ebene 2* einen anderen Aufbau als die zuvor beschriebene Vorlage der *Funktionsmuster.* Der Aufbau der *Verteilungsmustervorlage* wird im Nachfolgenden vorgestellt.

Musterkategorie: Die *Musterkategorie* vermittelt knapp und präzise die Art des Entwurfsmusters und ist für das schnelle Identifizieren und Einordnen des Problemlösungskontexts notwendig. In den Verteilungsmustern besitzt die *Musterkategorie* immer den Wert »Verteilungsmuster«, sodass ersichtlich ist, dass dieses Entwurfsmuster zur Lösung der Problemstellungen auf *Ebene 2* eingeordnet werden kann.

Mustertyp: Der *Mustertyp* beschreibt knapp und präzise die funktionale Anforderung, für die das Entwurfsmuster hilfreich ist. Durch den *Mustertyp* ist für den Anwendungsentwickler schnell ersichtlich, welche Entwurfsmuster für diese Anlagenfunktion hilfreich sind.

Mustername: Der *Mustername* vermittelt knapp und präzise den wesentlichen Inhalt des Entwurfsmusters. Wie Gamma in den Entwurfsmustern der Softwaretechnik erläutert, beschreibt der *Mustername* das Entwurfsproblem und seine Lösungen mit wenigen Worten [vgl. Gamma et al. 2004, S. 3]. Die Benennung von Mustern mithilfe eines *Musternamens* ist wichtig, weil dieser die Verteilungsart sowie die dazugehörenden Automatisierungsfunktionen beschreibt, für die dieses Entwurfsmuster geeignet ist.

Zweck: Der *Zweck* beschreibt das Grundprinzip sowie Sinn und Ziel des Entwurfsmusters und wird benötigt, um zu beschreiben, welche spezifischen Probleme und Fragestellungen das Entwurfsmuster behandelt.

Kontext: Der *Kontext* beschreibt den Anwendungs- und Nutzungskontext, um zu vermitteln, in welchen Situationen das Entwurfsmuster angewendet werden kann. Des Weiteren beschreibt der *Kontext* die Problemsituation, in der das Entwurfsmuster helfen kann.

Lösung: Die grafische Darstellung der *Lösung* ermöglicht eine Repräsentation der Verteilung der Automatisierungsfunktionen auf die Hardware sowie ihre Beziehungen und Interaktionen. Basierend auf Gamma, beschreibt die *Lösung* keinen bestimmten Entwurf und auch keine konkrete Implementierung, sondern lediglich eine Schablone, die in vielen verschiedenen Situationen angewendet werden kann [vgl. Gamma et al. 2004, S. 4]. Die *Lösung* ermöglicht eine abstrakte Beschreibung des Entwurfsproblems – Verteilung der Automatisierungsfunktionen — und zeigt auf, wie eine allgemeine Anordnung von Elementen – Automatisierungsfunktionen und Hardware – dieses Problem lösen kann [vgl. Gamma et al. 2004, S. 4]. Im Fall der *Verteilungsmuster* stellen die Automatisierungsfunktionen und die Hardware die verwendeten Elemente dar.

Teilnehmer: Dieser Abschnitt beschreibt textuell die Lösung des Entwurfsmusters, um dem Anwendungsentwickler ein einheitliches Verständnis zu vermitteln.

Konsequenzen: Der *Konsequenzabschnitt* beinhaltet die Auswirkungen der vorgeschlagenen Verteilungslösung auf die nfAs in tabellarischer Form. Durch diese Darstellung ist erkennbar, dass sich nfAs tendenziell verbessern oder verschlechtern können. Hierbei handelt es sich lediglich um die Vorstellung des Entwurfsmusters nicht um dessen Anwendung.

Vorteile: In diesem Abschnitt werden die Vorteile der Anwendung dieses Entwurfsmusters in Bezug auf weitere nfAs beschrieben. Dadurch ist ersichtlich, warum gerade dieses Entwurfsmuster verwendet werden sollte.

Nachteile: An dieser Stelle werden die Nachteile der Anwendung dieses Entwurfsmusters in Bezug auf weitere nfAs beschrieben. Dadurch ist ersichtlich, warum dieses Entwurfsmuster gegebenenfalls nicht verwendet werden sollte.

Eine ausschließlich grafische Repräsentation des Entwurfsmusters ist nicht ausreichend, da diese lediglich das Entwurfsergebnis darstellt [vgl. Gamma et al. 2004, S. 8]. Um Wiederverwendung in Form von Entwurfsmustern anzuwenden, müssen nicht nur die gefällten Entscheidungen aufgezeigt werden, sondern auch der Zweck sowie der Anwendungs- und Nutzungsbereich des

Entwurfsmusters. Die Entwurfsmustervorlage der Softwaretechnik wurde durch Zusammenfassen beziehungsweise Weglassen von Strukturelementen an die Problemstellungen sowie die Erfordernisse in Bezug auf verteilte Automatisierungssysteme angepasst (siehe Abbildung 7.5 auf der nächsten Seite).

Durch die zuvor beschriebene einheitliche Struktur wird das Verstehen, Vergleichen sowie Verwenden der unterschiedlichen Entwurfsmuster erleichtert [vgl. Gamma et al. 2004, S. 8]. Da die *Funktionsmuster* lediglich die Automatisierungsfunktionen beschreiben, die zur Erfüllung der Anlagenfunktion notwendig sind, und keine Implementierung beziehungsweise kein internes Verhalten der Funktionen, wurden diese nicht mit in die Entwurfsmustervorlage aufgenommen. Die einzelnen Interaktionen zwischen den Automatisierungsfunktionen sind anhand der Darstellung der Lösung ersichtlich. Deshalb wurde der Punkt *Interaktionen* der Softwaretechnik nicht mit in die *Funktionsmustervorlage* aufgenommen.

Da die zuvor beschriebene Vorlage für *Verteilungsmuster* einen sehr starken Bezug zu den nfAs hat, wurden in dieser ein Konsequenzabschnitt sowie die Vor- und Nachteile des Entwurfsmusters hinzugefügt. Eine Beschreibung der Entwurfsmuster ohne diese drei Abschnitte wäre nicht ausreichend zur Erfüllung der Herausforderungen auf *Ebene 2.*

Da die *Verteilungsmuster* lediglich die Verteilung von Automatisierungsfunktionen auf Knoten sowie die Berücksichtigung von nfAs beschreiben und keine Implementierung und kein internes Verhalten der Funktionen, wurden diese nicht mit in die Entwurfsmustervorlage aufgenommen. Die einzelnen Interaktionen zwischen den Automatisierungsfunktionen sind anhand der Darstellung der Lösung ersichtlich. Deshalb wurde der Punkt *Interaktionen* der Softwaretechnik nicht mit in die *Verteilungsmustervorlage* übernommen. Abbildung 7.5 ermöglicht einen Vergleich der Entwurfsmustervorlage der Softwaretechnik mit den zuvor beschriebenen Vorlagen der Funktions- und Verteilungsmuster.

7.2 Einbindung nicht-funktionaler Anforderungen in die Entwurfsmuster

Nachdem in den vorherigen Kapiteln sowohl die Relevanz der nfAs und die Entwurfsmustervorlage beschrieben wurden, werden in den nachfolgenden Abschnitten die Einbindung von nfAs in die Entwurfsmuster erläutert. Auf Basis des zuvor vorgestellten Merkmalskonzepts erfolgt eine Darstellung der nfAs in den Entwurfsmustern sowie eine Beschreibung, wie die Erfüllung beziehungsweise Nichterfüllung von nfAs berechnet wird.

7.2.1 Darstellung von nicht-funktionalen Anforderungen in den Entwurfsmustern

Die Erfüllung von nfAs ist abhängig von der Verteilung der Automatisierungsfunktionen im System. Deshalb ist die Berücksichtigung von nfAs gerade auf *Ebene 2* des Vorgehensmodells von großer Bedeutung; die nfAs müssen in die *Verteilungsmuster* eingebunden werden. Hierzu ist es notwendig, unterschiedliche Darstellungsarten zu bewerten und zu evaluieren. Die erste Darstellungsart ist die *grafische Darstellung der nfAs in den Entwurfsmustern.* Hierzu wurden insgesamt sieben mögliche grafische Darstellungen von nfAs in den Entwurfsmustern aufgezeigt und bewertet. Aufgrund der Vielzahl von Merkmalen – Anforderungs- und Lösungsmerkmale – in den nfAs sowie der inhaltlichen Zusammengehörigkeit dieser Merkmale ist eine grafische Darstellung von nfAs in den Entwurfsmustern sehr unübersichtlich. Deshalb wurde der grafische Ansatz der Einbindung von nfAs nicht weiterverfolgt. Der zweite Ansatz einer möglichen

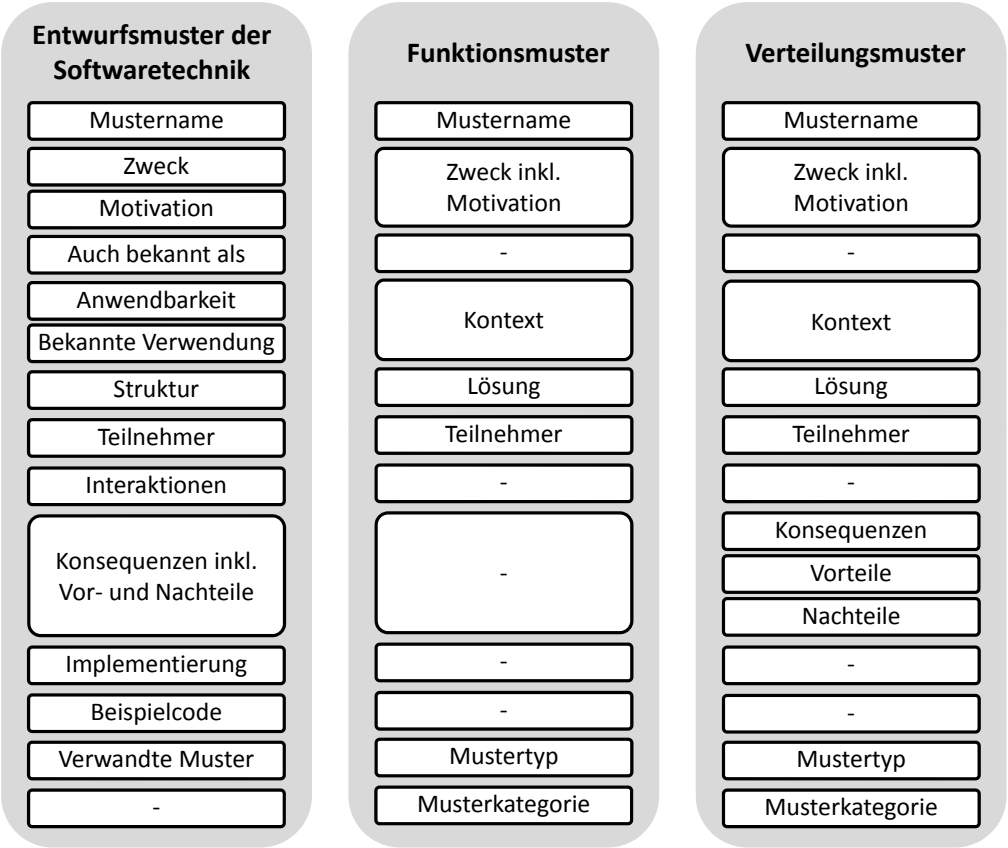

Abbildung 7.5: Vergleich der Entwurfsmustervorlagen

Darstellung ist die *tabellarische Darstellung in den Verteilungsmustern.* Da der Anwendungsentwickler durch eine grafische Einbindung von nfAs keine optimale Entwurfsunterstützung erhält, wurde eine tabellarische Darstellung von nfAs in den Entwurfsmustern entworfen. Bei der Darstellung der nfAs muss einerseits zwischen nfAs in den Entwurfsmustern und andererseits zwischen Darstellung der nfAs bei Anwendung der Verteilungsmuster unterschieden werden.

Die tendenziellen Auswirkungen der nfAs – Darstellung in den Entwurfsmustern – sowie die Auswirkungen der nfAs – Anwendung des Entwurfsmusters – werden in der tabellarischen Darstellung der nfAs in der Spalte »Trend« grafisch dargestellt (siehe Tabelle 7.1 und 7.2 auf Seite 78). Durch einfache und allgemeingültige grafische Symbole sind die jeweiligen Auswirkungen der infrage stehenden Verteilungsalternative auf einen Blick ersichtlich. Bei der Darstellung der nfAs in den Entwurfsmustern erfolgt im Konsequenzabschnitt des Verteilungsmusters die Beschreibung der Anforderungs- und Lösungsmerkmale sowie der tendenziellen Auswirkungen der Anwendung dieses Musters auf die Lösungsmerkmale in tabellarischer Form. Die tabellarische Darstellung (siehe Tabelle 7.2) beinhaltet die Anforderungs- und Lösungsmerkmale und gibt einen Überblick hinsichtlich der tendenziellen Auswirkungen für jeweils ein Merkmal der nfAs. Auf Basis der Werte der Anforderungsmerkmale, die auf Grundlage der Ausgangsdokumente (zum Beispiel Topologiemodell) der vorherigen Ebene vorliegen, werden bei Anwendung des Verteilungsmusters die Lösungsmerkmale ermittelt. Durch die tabellarische Darstellung der Merkmale und der ermittelten Werte der Lösungsmerkmale ist ersichtlich, ob die nfAs erfüllt sind oder nicht (siehe Tabelle 7.2).

Tabelle 7.1: Symbole des Konsequenzabschnitts

Symbol	Bedeutung
✔	Dieses Symbol steht für: positive tendenzielle Auswirkungen der Anwendung dieses Entwurfsmusters auf das Lösungsmerkmal der nfA
✘	Dieses Symbol steht für: negative tendenzielle Auswirkungen der Anwendung dieses Entwurfsmusters auf das Lösungsmerkmal der nfA
!	Dieses Symbol steht für: Warnung, hier können negative Auswirkungen auf das Lösungsmerkmal der nfA auftreten.

Tabelle 7.2: Darstellung der Anforderungs- und Lösungsmerkmale in den Verteilungsmustern

Anforderungsmerkmale		Lösungsmerkmale			
Merkmals-träger	Ausprägung	Trend	Merkmals-träger	Berechnung	Ausprägung
Merkmal 1		Merkmal 1			
Knoten 1	Diese Daten werden automatisch ausgefüllt.	✔	Knoten 1	Berechnungs-formel	Ergebnis
Merkmal 2		Merkmal 2			
Knoten 2	Diese Daten werden automatisch ausgefüllt.	✘	Knoten 2	Berechnungs-formel	Ergebnis

7.2.2 Erfüllungsgrad und Zusammenspiel relevanter nicht-funktionaler Anforderungen

Um feststellen zu können, ob eine nfA erfüllt ist, müssen hierzu Eigenschaften beziehungsweise Kriterien gefunden werden, die valide Aussagen ermöglichen. Dies bedeutet, eine nfA muss messbar und quantifizierbar gemacht werden. Basierend auf der in Abschnitt 7.2.1 beschriebenen Darstellung der nfAs in den Entwurfsmustern und dem Merkmalskonzept, das die nfAs messbar und quantifizierbar macht, kann der Erfüllungsgrad der nfAs für die Verteilungsalternativen ermittelt werden.

Aufgrund der zugrunde liegenden Anforderungsmerkmale werden die Lösungsmerkmale der nfAs berechnet, sodass für den Anwender ersichtlich ist, ob das ausgewählte Entwurfsmuster die relevanten nfAs erfüllt. Dieser Ansatz ermöglicht es, den Anwendungsentwickler mit einer Vorgehensweise zu führen sowie auch bei Designentscheidungen jeweils die Auswirkungen der Entscheidung abzuschätzen und den Anwendungsentwickler bei den Entscheidungen zu unterstützen. So kann die Erfüllung von nfAs schon während des Entwurfs berücksichtigt werden.

Das Entwurfsergebnis, das die für die Anwendung notwendigen Funktionsblöcke und deren Laufzeitumgebung enthält, kann dann in einer der SPS-Programmiersprachen der IEC 61 131-3 programmiert werden.

Bei Anwendung des Verteilungsmusters wird die Tabelle der Anforderungsmerkmale automatisiert ausgefüllt, und die Lösungsmerkmale werden berechnet, sodass der Erfüllungsgrad der nfAs ersichtlich ist. Bei Nichterfüllung der Lösungsmerkmale wird zusätzlich zur tabellarischen Darstellung der Knoten die Verbindung rot hinterlegt, sodass direkt im Modell ersichtlich ist, dass das Lösungsmerkmal nicht erfüllt wird. Die nachfolgende Tabelle 7.3 zeigt exemplarisch die Berechnung der nfA *Ressourcennutzung.* Auf Grundlage der Anforderungsmerkmale *Bereitgestellter Speicher* und *Benötigter Speicher* wird das Lösungsmerkmal *Speicherkompatibilität* berechnet.

Tabelle 7.3: Berechnung der nfA Ressourcennutzung – Lösungmerkmal Speicherkompatibilität

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
		Speicherkompatibilität (load)	
✔	Knoten 1	Bereitgestellter Speicher (load) ↔ Benötigter Speicher (load)	`IF` Bereitgestellter Speicher (load) < Benötigter Speicher (load) `THEN` false `ELSE` true
		Speicherkompatibilität (runtime)	
✔	Knoten 1	Bereitgestellter Speicher (runtime) ↔ Benötigter Speicher (runtime)	`IF` Bereitgestellter Speicher (runtime) < Benötigter Speicher (runtime) `THEN` false `ELSE` true

Die Merkmale unterschiedlicher nfAs können sich untereinander positiv oder negativ beeinflussen. Als Beispiele für eine negative Beeinflussung sind das Merkmal »Zahl der Submodule« der nfA Modularität sowie das Merkmal »Durchlaufzeit« der nfA Zeitverhalten zu nennen, weil die Durchlaufzeit beispielsweise von der Anzahl der Submodule abhängig ist und durch eine steigende Anzahl von Submodulen die Durchlaufzeit negativ beeinflusst wird. Diese Konflikte der nfAs müssen erkannt und gelöst werden, was durch die Anwendung der Entwurfsmuster unterstützt wird.

7.3 Verwendete Darstellungsform in den Entwurfsmustern

Die nachfolgenden Abschnitte erläutern das verwendete Beschreibungsmittel zur Darstellung der Lösung in den Funktions- und Verteilungsmustern. Das Beschreibungsmittel basiert auf der vom AIS entwickelten und in [Frank et al. 2013b] beschriebenen Notation. Diese Notation basiert auf Elementen der SysML und wird zur Darstellung der Lösung verwendet, da diese speziell zur Modellierung von verteilten Automatisierungssystemen entwickelt wurde und somit alle relevanten Aspekte abdeckt.

7.3.1 Funktionsmuster

Wie in Abschnitt 7.1.2 beschrieben, kann das *Funktionsmuster* aufgrund der in Abschnitt 6.1.2 erläuterten Aktivitäten auf *Ebene 4* eingeordnet werden. Aufgrund dieser Einordnung besteht, wie in Abschnitt 7.1.1 beschrieben, ein *Funktionsmuster* in seiner Darstellung aus einer Anlagenfunktion sowie einer Vielzahl von Automatisierungsfunktionen. Diese Elemente müssen in den Entwurfsmustern grafisch dargestellt werden (siehe Tabelle 7.4 auf Seite 81), um dem Anwender eine einfache Repräsentation und Nachverfolgung der Lösung zu ermöglichen.

Die Anlagenfunktion wird mithilfe des Notationselements *«Anforderung»* dargestellt und beinhaltet im Anforderungstyp den Wert *Anlagenfunktion* [vgl. Frank et al. 2013b, S. 7]. Da eine Anlagenfunktion durch ein oder mehrere Automatisierungsfunktionen realisiert wird, muss diese Anforderungsgültigkeit durch eine gestrichelte Linie dargestellt werden [vgl. Frank et al. 2013b, S. 7]. Im Falle der Darstellung der Lösung in den *Funktionsmustern* ist eine Anforderungsgültigkeit eine Beziehung zwischen einer Anlagenfunktion – dargestellt durch das Notationselement *«Anforderung»* – und einer oder mehrerer Automatisierungsfunktion(en) – dargestellt durch das Notationselement *«Funktion».*

Die in den Entwurfsmustern dargestellten Automatisierungsfunktionen können weitere Automatisierungsfunktionen beinhalten und sind Grundlage für das funktionale Verhalten der Anlage. Die Darstellung der Automatisierungsfunktionen beschreibt somit die funktionale Struktur sowie die logischen Zusammenhänge der funktionalen Anforderung [vgl. Frank et al. 2013b, S. 9]. Diese Automatisierungsfunktionen werden in späteren Entwurfsphasen in Funktionsblöcke unterteilt, die in den Programmiersprachen der IEC 61 131-3 implementiert werden können. Wegen der engen Kopplung und Kommunikation zwischen den einzelnen Automatisierungsfunktionen wird nach zyklischen – dargestellt durch eine durchgezogene Linie – und azyklischen Verbindungen – dargestellt durch eine gestrichelte Linie – zwischen den Funktionen unterschieden [vgl. Frank et al. 2013b, S. 9]. Während die zyklische Verbindung einen späteren Austausch von Parametern und Messwerten darstellt, zeigt die azyklische Verbindung eine logische Abfolge von Funktionen [vgl. Frank et al. 2013b, S. 9]. Die zuvor beschriebenen Notationselemente finden in unterschiedlichen Kombinationen in den *Funktionsmustern* Anwendung. Abbildung 7.6 zeigt eine anwendungsneutrale Darstellung einer möglichen Lösung in den Funktionsmustern.

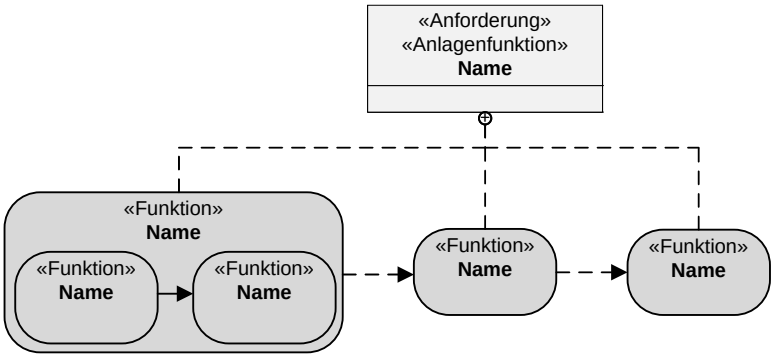

Abbildung 7.6: Anwendungsneutrale Darstellung der Lösung eines Funktionsmusters

Das Notationskonzept ist für die Darstellung und Erarbeitung des Entwurfsergebnisses verteilter Automatisierungssysteme hilfreich und ausreichend, benötigt für die Darstellung der Entwurfsmusterlösung jedoch weitere Elemente, die in diesem Ansatz nicht vorgesehen sind. In der Darstellung von Lösungen für Entwurfsmuster wird im Gegensatz zur Darstellung der Entwurfslösung eine mögliche Auswahl von alternativen Automatisierungsfunktionen benötigt, sodass der Anwendungsentwickler bei Bedarf eine Automatisierungsfunktion durch eine Auswahl von Automatisierungsfunktionen näher spezifiziert. Hierzu kann der Anwendungsentwickler aus mehreren Automatisierungsfunktionen eine oder gegebenenfalls mehrere auswählen.

Für die zuvor beschriebene Problemstellung existieren eine Vielzahl von Ansätzen, wie beispielsweise aus dem Bereich der Informatik [Rupp et al. 2007], GRAphe Fonctionnel de Commande Etape Transition (grafcet) [IEC 60 848 2013]# und in der Betriebswirtschaftslehre (bwl) [Nüttgens & Rump 2002]. Die Automatisierungstechnik ermöglicht in ihren Modellie-

Tabelle 7.4: Notationselemente der Funktionsmuster [vgl. Frank et al. 2013b, S. 8 f.]

Notationselement	Bedeutung
«Anforderung» «Anlagenfunktion» **Name**	Notationselement *Anlagenfunktion*
«Funktion» **Name**	Notationselement *Automatisierungsfunktion*
«Funktion» **Name** «Funktion» **Name** «Funktion» **Name**	Notationselement *Automatisierungsfunktion mit untergliederten Automatisierungsfunktionen*
	Notationselement *Zyklische Verbindung*
	Notationselement *Azyklische Verbindung*
	Notationselement *Anforderungsgültigkeit*

rungssprachen, wie beispielsweise Petri-Netze [ISO/IEC 15 909-1 2004][#], eine Parallel- oder Alternativverzweigung. Diese werden aufgrund der Vielzahl von spezifischen Automatisierungsfunktionen nicht näher betrachtet, da die Darstellung der Entwurfsmusterlösung feingranular ist, vom Anwendungsentwickler näher spezifiziert werden muss und deshalb keine geeignete übersichtliche Darstellung ermöglicht.

Die Symbole des GRAFCET und die Symbole der BWL für »UND« sowie »ODER« haben in der Automatisierungstechnik eine festgelegte Bedeutung und werden deshalb nicht für die oben erläuterte nähere Spezifikation von Automatisierungsfunktionen verwendet. Deshalb wird für die Darstellung von »exklusivem ODER« sowie »ODER« keine neue Symbolik eingeführt, sondern diese grafisch durch einen Kreis dargestellt (siehe Abbildung 7.7 auf der nächsten Seite).

Da in der Musterlösung durch die zyklischen und azyklischen Verbindungen logische Abfolgen dargestellt werden, wird eine separate Darstellung von »UND« nicht benötigt. Die Darstellung eines »XOR« zeigt, dass aus den jeweils damit verbundenen Automatisierungsfunktionen eine zur näheren Spezifikation ausgewählt werden muss. Während die Darstellung des »XOR« die Auswahl von nur einer Automatisierungsfunktion erlaubt, ermöglicht die »OR«-Darstellung eine Auswahl von mehreren Automatisierungsfunktionen.

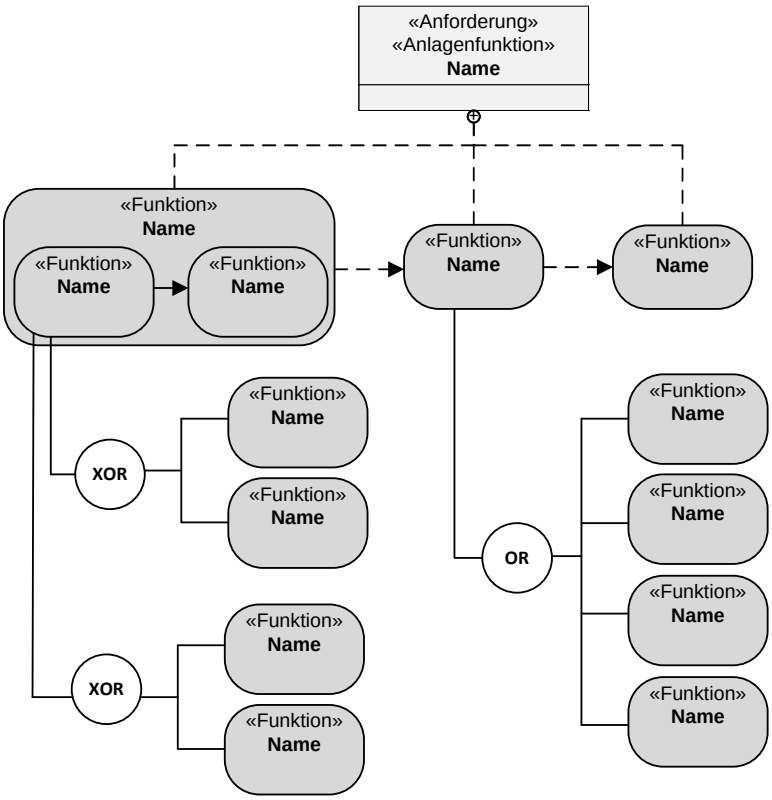

Abbildung 7.7: Darstellung »exklusives ODER« und »ODER« in den Funktionsmustern

7.3.2 Verteilungsmuster

Wie in Abschnitt 7.1.2 beschrieben, kann das *Verteilungsmuster* aufgrund der in Abschnitt 6.1.2 erläuterten Aktivitäten auf *Ebene 2* eingeordnet werden. Die *Ebene 2* beschreibt die Verteilung der auf *Ebene 3* spezifizierten Funktionsblöcke, die aus den Automatisierungsfunktionen abgeleitet wurden. Diese Funktionsblöcke beinhalten das spätere interne Verhalten der Funktionen und können miteinander über Schnittstellen (Ports) kommunizieren. Aufgrund dieser Einordnung und der darin enthaltenen Aktivitäten besteht, wie in Abschnitt 7.1.1 beschrieben, ein *Verteilungsmuster* in seiner Darstellung aus einer Vielzahl von Funktionsblöcken, Knoten, automatisierungstechnischen Geräten, Ports und Verbindungen. Diese Elemente müssen in den Entwurfsmustern grafisch dargestellt werden (siehe Tabelle 7.5 auf der nächsten Seite), um dem Anwender eine einfache Repräsentation und Nachverfolgung der Lösung zu ermöglichen.

Die benötigte Hardware (Steuerung, Sensor oder Aktor) wird mithilfe des Notationselements *«Knoten»* dargestellt und beinhaltet im Knotentyp entweder den Wert *Aktor, Sensor* oder *Steuerung* [vgl. Frank et al. 2013b, S. 10 f.]. Die Funktionalität der Anlage wird durch die Funktionsblöcke – dargestellt durch das Notationselement *«FB»* – realisiert [vgl. Frank et al. 2013b, S. 9]. Diese Funktionsblöcke müssen in *Ebene 2* auf die jeweilige Hardware verteilt werden. Um die Funktionalität der Anlage zu realisieren, müssen die in den Entwurfsmustern dargestellten Funktionsblöcke untereinander sowie mit der Hardware kommunizieren. Deshalb wird zwischen Ausgangsport, Eingangsport, Datenverbindung und Steuerverbindung unterschieden [vgl. Frank et al. 2013b, S. 10]. Die Ports sind notwendig, um eine Kommunikation zwischen diesen Elementen zu ermöglichen, und repräsentieren einen Ausgabe- oder Eingabepa-

rameter. Des Weiteren wird zwischen Datenverbindung – dargestellt durch eine durchgezogene Linie – und Steuerverbindung – dargestellt durch eine gestrichelte Linie – unterschieden (siehe Tabelle 7.5), die mithilfe der Ports eine Kommunikation ermöglichen [vgl. Frank et al. 2013b, S. 10]. Durch eine Datenverbindung werden Informationen übermittelt, wie zum Beispiel der Austausch von Messwerten. Eine Steuerverbindung leitet Steuersignale weiter, wie zum Beispiel das Steuersignal an einen Aktor.

Tabelle 7.5: Notationselemente der Verteilungsmuster [vgl. Frank et al. 2013b, S. 10 f.]

Notationselement	Bedeutung
«FB» **Name**	Notationselement *Funktionsblock*
	Notationselement *Ausgangsport*
	Notationselement *Eingangsport*
	Notationselement *Datenverbindung*
	Notationselement *Steuerverbindung*
«Knoten» «Knotentyp» **Name : Gerätetyp**	Notationselement *Hardware*

Die zuvor beschriebenen Notationselemente finden in unterschiedlichen Kombinationen in den Lösungen der *Verteilungsmuster* Anwendung. Abbildung 7.8 zeigt eine anwendungsneutrale Darstellung einer möglichen Lösung in den Verteilungsmustern.

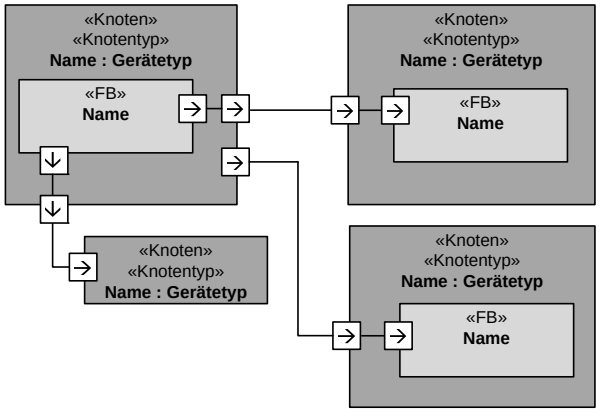

Abbildung 7.8: Anwendungsneutrale Darstellung der Lösung eines Verteilungsmusters

7.4 UML-Modell für die Anwendung der Entwurfsmuster

Das zuvor beschriebene Konzept der Entwurfsunterstützung besteht aus zwei verschiedenen Entwurfsmustern, nämlich Funktions- und Verteilungsmuster. Beide Musterarten können in unterschiedliche Sichten untergliedert werden, und zwar in die *Programmsicht* (M2 – Metamodellschicht), die *Mustersicht* (M1 – Modellschicht) und die *Anwendungssicht* (M0 – Laufzeitschicht). Die zuvor genannten Sichten können in die 4-Schichtenarchitektur der Object Management Group (OMG) [OMG][@] eingeordnet werden. Für jede dieser Sichten werden die zu verwendenden Elemente festgelegt, sodass eine jeweils auf die Sichten (Programmsicht, Mustersicht, Anwendungssicht) bezogene Darstellung der Entwurfsmuster ermöglicht wird. Da für die Darstellung der Entwurfsmuster lediglich drei Sichten benötigt werden, wird die Meta-Metamodellschicht weggelassen. Abbildung 7.9 zeigt die drei Sichten auf das Entwurfsmusterkonzept.

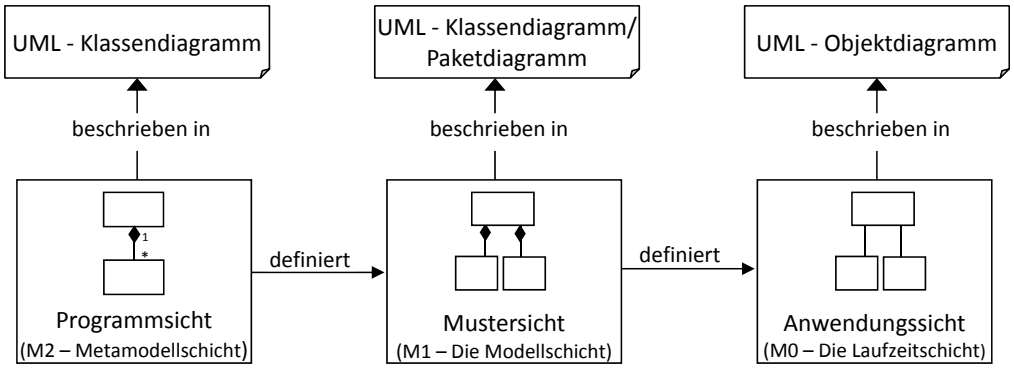

Abbildung 7.9: Sichten auf die Muster

In der *Programmsicht* wird der allgemeingültige Aufbau eines Entwurfsmusters dargestellt. Hierzu wird der grundlegende Aufbau des Entwurfsmusters in Form von Klassen dargestellt, die über Vererbung, Komposition, Aggregation sowie Assoziation miteinander verbunden sind. Als Beschreibungsmittel für diese Sicht und für statische Strukturen wird grundsätzlich das UML-Klassendiagramm verwendet. Die Programmsicht beschreibt die Definition eines Entwurfsmusters und enthält die darin enthaltenen Klassen. Des Weiteren enthält diese Sicht die Syntax (Regeln) für den Aufbau eines Entwurfsmusters und ist somit Grundlage für die Mustersicht und für das Hinzufügen neuer Entwurfsmuster.

In der *Mustersicht* wird der statische Aufbau eines konkreten Entwurfsmusters dargestellt. Hierzu werden die konkreten Elemente des Entwurfsmusters und deren Relationen beschrieben. Im Gegensatz zur Anwendungssicht beschreibt diese Sicht den grundlegenden Aufbau eines konkreten Entwurfsmusters und keine Instanzen. Als Beschreibungsmittel für dieses UML-Modell kann das UML-Paketdiagramm, das die Klassen und Assoziationen beinhaltet, verwendet werden.

Die *Anwendungssicht* beschreibt eine Instanz des Entwurfsmusters zur Laufzeit. In dieser Sicht wird das Entwurfsmuster aus einer Vielzahl von Objekten dargestellt, die über Assoziationen miteinander verbunden sind. Als Beschreibungsmittel für dieses UML-Modell kann das UML-Objektdiagramm verwendet werden.

Diese drei Sichten sind nicht unabhängig voneinander, sondern bauen aufeinander auf (siehe Abbildung 7.10 auf der nächsten Seite). Während die *Programmsicht* den grundsätzlichen Aufbau eines Entwurfsmusters darstellt, beschreibt die *Mustersicht* den Aufbau eines konkreten Entwurfsmusters und die *Anwendungssicht* die Instanz dieses konkreten Entwurfsmusters. Die nachfolgende Abbildung 7.10 beschreibt den grundsätzlichen Zusammenhang dieser drei

Sichten. Der grundlegende Aufbau und Zusammenhang des Entwurfsmusterkonzepts lässt sich in die 4-Schichtenarchitektur der OMG einordnen (siehe Abbildung 7.11). Abbildung 7.10 und Abbildung 7.11 zeigen beide den Zusammenhang der zuvor beschriebenen Sichten. Die Einordnung der Sichten der Entwurfsmuster in die 4-Schichtenarchitektur (siehe Abbildung 7.11) erfolgt in die Schichten M2 – Metamodellschicht, M1 – Modellschicht und M0 – Laufzeitschicht.

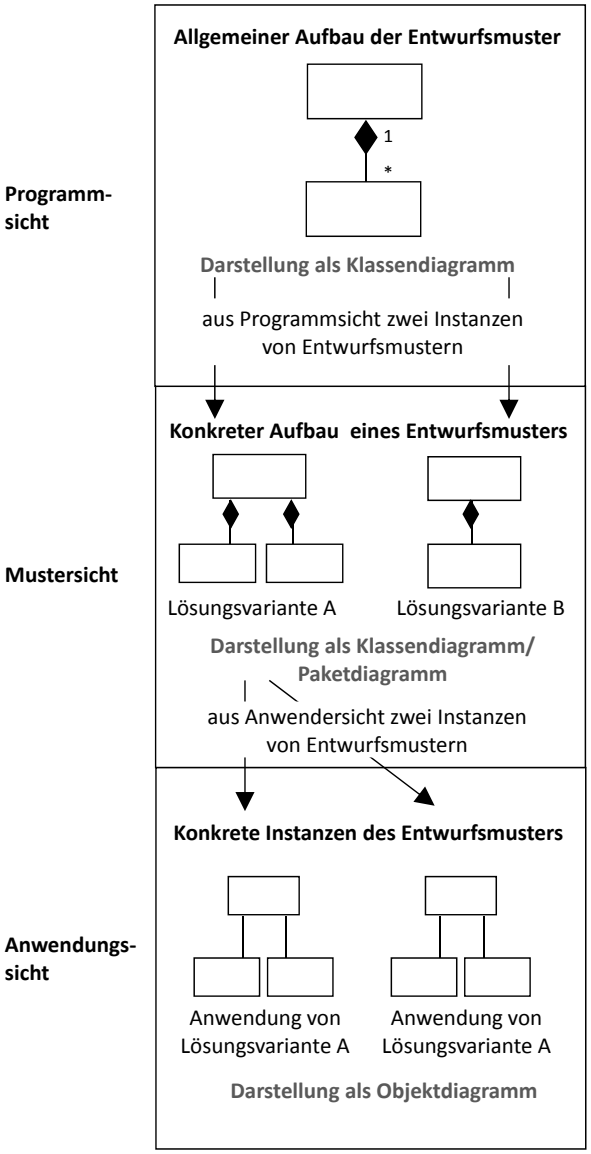

Abbildung 7.10: Zusammenhang der Sichten

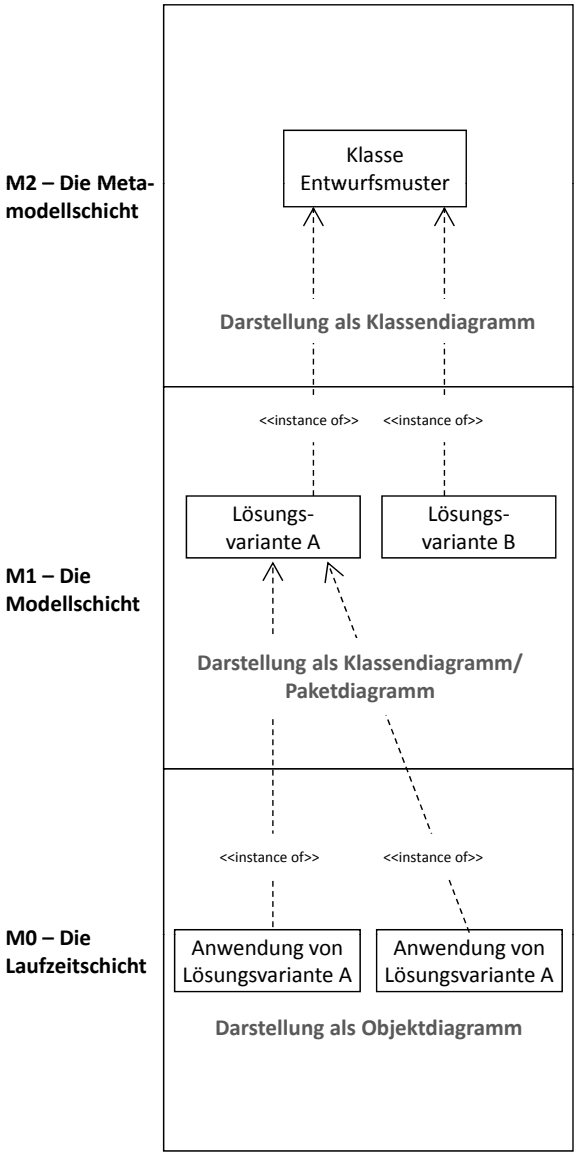

Abbildung 7.11: Zusammenhang der drei Sichten (M0–M2), basierend auf der 4-Schichtenarchitektur

8 Beispiele für Entwurfsmuster verteilter Automatisierungssysteme

Nachdem in Kapitel 7 das Konzept der Entwurfsmuster sowie die Einordnung von nfAs in dieses aufgezeigt wurden, erfolgt im nachfolgenden Kapitel eine beispielhafte Darstellung des Entwurfsmusterkonzepts anhand je eines *Funktions-* und *Verteilungsmusters* für die Anlagenfunktion *Transportieren*. Die Anlagenfunktion *Transportieren* findet in unterschiedlichen Automatisierungssystemen Anwendung (beispielsweise in einem Transportsystem zwischen zwei Maschinen oder in einer verfahrenstechnischen Anlage zum Transportieren von Flüssigkeiten). Die Entwurfsmuster werden auf Basis der in Abschnitt 7.1.3 vorgestellten Entwurfsmustervorlagen beschrieben. Anschließend werden sie mithilfe von UML-Diagrammen – entsprechend Abschnitt 7.4 – dargestellt.

8.1 Ableitung eines Funktionsmusters

Das folgende Beispiel eines Entwurfsmusters beschreibt ein mögliches *Funktionsmuster* für die zuvor genannte Aufgabenstellung eines Transportsystems und beinhaltet die Anlagenfunktion *Transportieren*. Für sie können eine Vielzahl möglicher *Funktionsmuster* ausgewählt werden, die sich in den Automatisierungsfunktionen unterscheiden. Basierend auf den Projektanforderungen, die zumeist in einem Anforderungsdokument der Automatisierungstechnik, wie beispielsweise Lastenheft oder Anlagenlayout, festgehalten werden, zeigt das nachfolgende *Funktionsmuster* eine Möglichkeit der Umsetzung dieser Anforderung. Das *Funktionsmuster* kann, wie zuvor beschrieben, auf *Ebene 4* Anwendung finden. Es zeigt eine mögliche Ausprägung für die zuvor genannte Problemstellung eines Transportsystems und beinhaltet die Wegverfolgung und die Reservierung. Dieses Entwurfsmuster wird in Kapitel 9 wieder aufgegriffen und angewendet.

Musterkategorie: Funktionsmuster

Mustertyp: Transport

Mustername: Transportieren von Komponenten inklusive Wegverfolgung und Reservierung

Zweck: Dieses Entwurfsmuster beinhaltet alle relevanten Automatisierungsfunktionen – inklusive Wegverfolgung (Weg suchen und Weg wählen) und Reservierung (Weg reservieren und Reservierung aufheben) –, um die Anlagenfunktion *Transportieren* zu erfüllen.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Dieses Transportmuster ist anzuwenden, wenn

- für den Transport mehrere Transportwege zur Verfügung stehen.
- eine Reservierung von Ressourcen benötigt wird, damit diese von mehreren Anlagenfunktionen genutzt werden können.

Lösung: Die nachfolgende Abbildung (siehe Abbildung 8.1 auf der nächsten Seite) zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen in jedem SPS-Zyklus.

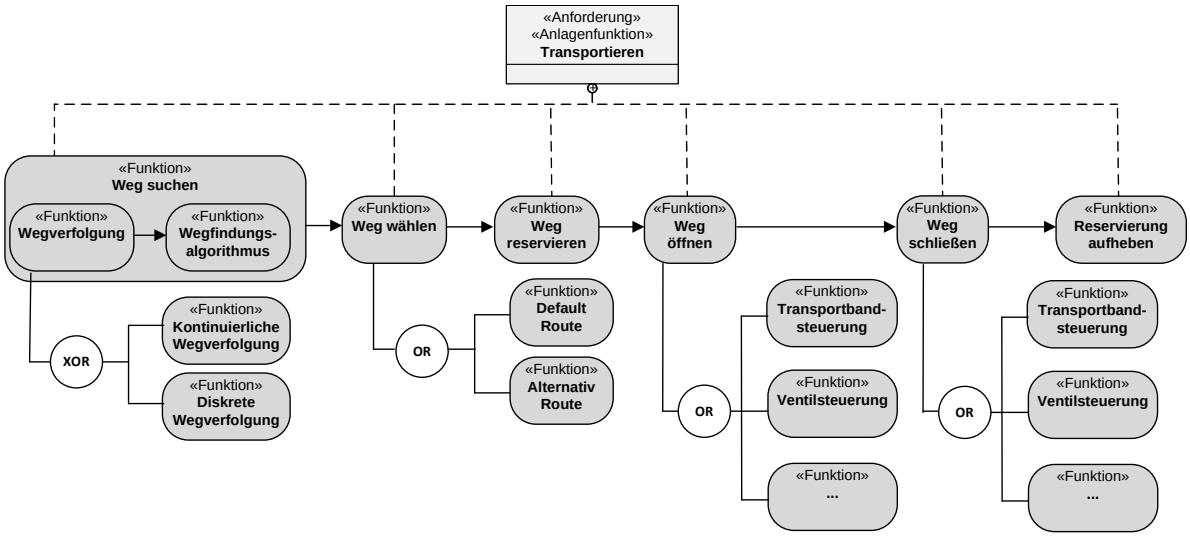

Abbildung 8.1: Funktionsmuster Transportieren

Teilnehmer: Dieses Funktionsmuster besteht grundsätzlich aus sechs Automatisierungsfunktionen, die im Folgenden näher beschrieben werden:

1. *Weg suchen:* Für einen Transport von Komponenten, wie beispielsweise Werkstücke oder Flüssigkeiten, stehen oftmals unterschiedliche Wege zur Verfügung. Zum Suchen geeigneter Transportwege stehen unterschiedliche Verfahren zur Verfügung. Zuerst wird mithilfe einer diskreten oder kontinuierlichen Wegverfolgung der aktuelle Standort der Komponente ermittelt. Im Nachfolgenden wird/werden auf Basis des Wegfindungsalgorithmus ein Transportweg oder mehrere Transportwege gesucht.
2. *Weg wählen*: Sobald verschiedene Transportwege gefunden wurden, muss der Transportweg, der zum Transport der Komponenten verwendet werden soll, ausgewählt werden. Hierzu stehen die Funktionen »Default Route« und »Alternativ Route« zur Verfügung.
3. *Weg reservieren:* Wurde ein geeigneter Transportweg ausgewählt, müssen mit dieser Automatisierungsfunktion die hierfür benötigten Ressourcen reserviert werden.
4. *Weg öffnen:* Nachdem die benötigten Ressourcen erfolgreich reserviert wurden, muss der Transportweg bei Bedarf geöffnet werden. Zum Öffnen oder Anschalten von Ressourcen, wie beispielsweise Anschalten von Förderbändern oder Öffnen von Ventilen, stehen mehrere Alternativen zur Verfügung. Hierbei ist zu beachten, dass die jeweiligen Ressourcen zum einen lediglich angeschaltet oder geöffnet und zum anderen mit bestimmten Parametern, wie beispielsweise Geschwindigkeit, angeschaltet oder geöffnet werden können.
5. *Weg schließen:* Nach erfolgreichem Transportieren der Komponenten muss der Transportweg geschlossen werden. Dies bedeutet, dass beispielsweise Förderbänder ausgeschaltet und Ventile geschlossen werden. Hierzu stehen die gleichen Verfahren wie beim Weg öffnen zur Verfügung.
6. *Reservierung aufheben:* Sobald der Transportweg geschlossen wurde, wird die Reservierung der Ressourcen wieder aufgehoben. Dieser Schritt ist notwendig, damit andere Aufträge ebenfalls diese Ressourcen zum Transportieren verwenden können.

Eine Variante des zuvor beschriebenen Entwurfsmusters ist zum Beispiel ein Funktionsmuster, das die Funktionen »Weg öffnen« und »Weg schließen« beinhaltet. Die Funktionsmuster sind inhaltlich auf den jeweiligen Kontext bezogen und unterscheiden sich in den Automatisierungsfunktionen. Deshalb ist ein Aufzeigen des Nutzens nicht Bestandteil von Funktionsmustern.

8.2 Ableitung eines Verteilungsmusters

Das nachfolgende Beispiel eines Entwurfsmusters beschreibt ein mögliches *Verteilungsmuster* für die zuvor erläuterte Problemstellung eines Transportsystems. Für diese Problemstellung existiert eine Vielzahl an *Verteilungsmustern,* die sich in der Verteilung der Funktionalitäten unterscheiden. Basierend auf den relevanten nfAs, die zumeist in einem Anforderungsdokument der Automatisierungstechnik festgehalten werden, zeigt das nachfolgende *Verteilungsmuster* eine Möglichkeit für die Verteilung der Funktionalität. Dieses *Verteilungsmuster* kann, wie zuvor beschrieben, auf *Ebene 2* Anwendung finden. Das nachfolgende *Verteilungsmuster* zeigt eine mögliche Verteilungsalternative der zuvor im *Funktionsmuster Transportieren* definierten Funktionalität.

Musterkategorie: Verteilungsmuster

Mustertyp: Verteiltes Transportieren

Mustername: Verteiltes Transportieren inklusive Wegverfolgung und Reservierung – Wegverfolgung und Reservierung zentral und separate Ressourcenansteuerung

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der oben genannten Automatisierungsfunktionen unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten. Die Automatisierungsfunktionen für die Wegverfolgung – Weg suchen und Weg wählen – und für die Reservierung – Weg reservieren und Reservierung aufheben – werden zentral realisiert.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungstechnik und in der Verfahrenstechnik. Dieses Transportmuster ist anzuwenden, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- mehrere Knoten zur Verfügung stehen, um die Funktionalität auf diese zu verteilen.

Lösung: Die nachfolgende Abbildung (siehe Abbildung 8.2 auf der nächsten Seite) zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird die Anlagenfunktion *Transportieren* auf zwei verschiedene Knoten verteilt. Die nachfolgende Lösung zeigt zwei nacheinander zu durchlaufende Transportwege sowie die Verteilung der dafür benötigten Funktionalität auf drei Knoten. Die Wegauswahl – Weg suchen und Weg wählen – und die Reservierung – Weg reservieren und Reservierung aufheben – wird für alle Transportwege zentral realisiert.

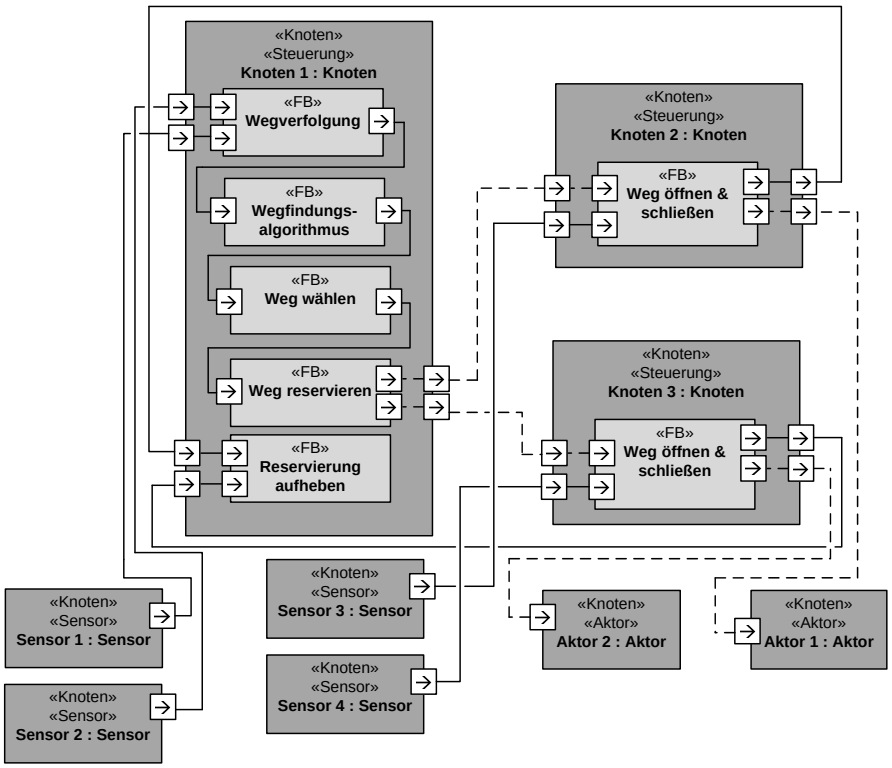

Abbildung 8.2: Verteilungsmuster Transportieren

Teilnehmer: Dieses Verteilungsmuster besteht in der Regel aus drei Knoten, vier Sensoren sowie zwei Aktoren, die im Folgenden näher beschrieben werden:

1. *Knoten 1:* Die Funktionalität der Wegauswahl – Weg suchen und Weg wählen – sowie der Reservierung – Weg reservieren und Reservierung aufheben – wird zentral auf einem Knoten für alle Transportwege realisiert.
2. *Knoten 2 und Knoten 3:* Die Ansteuerung der Ressourcen erfolgt getrennt auf jeweils einem separaten Knoten. Diese Knoten müssen mit dem Knoten 1 sowie den jeweiligen Aktoren kommunizieren.
3. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, die geöffnet beziehungsweise angeschaltet oder geschlossen beziehungsweise ausgeschaltet werden müssen.
4. *Sensor 1 und Sensor 2:* Die Sensoren geben Rückmeldung über den Standort des Transportguts am Beginn des Transports.
5. *Sensor 3 und Sensor 4:* Die Sensoren geben Rückmeldung über den Standort des Transportguts am Ende des Transports.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben (siehe Tabelle 8.1 und 8.2 auf Seite 90). Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich. Die Anforderungsmerkmale können aus den Gerätedatenblättern sowie den SPS-Datenblättern entnommen werden. Die Anforderungsmerkmale »Ausführungszeit einer Program Organization Unit (POU)« und »Benötigter Speicher« sind abhängig von den programmierten Funktionsblöcken.

Tabelle 8.1: Anforderungsmerkmale im Verteilungsmuster »Transportieren«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Die in der nachfolgenden Tabelle genannten Lösungsmerkmale und ihre Eigenschaften (Merkmalsträger, Berechnung und Ausprägung) können erst bei einer kontextbezogenen Anwendung des Verteilungsmusters und bei ausgefüllten Anforderungsmerkmalen berechnet werden. Bei der Berechnung und dem Vergleich der Berechnungsergebnisse mit den Anforderungen – Ausprägungen der Anforderungsmerkmale – ist ersichtlich, ob ein Lösungsmerkmal erfüllt ist oder nicht. Die Eigenschaft *Merkmalsträger* beschreibt beispielsweise auf welchen Knoten oder Sensor sich das Lösungsmerkmal bezieht.

Tabelle 8.2: Lösungsmerkmale im Verteilungsmuster »Transportieren«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden:

- Ressourcennutzung: Bei diesem Entwurfsmuster ist kein Speicher an jedem Knoten notwendig – Senkung der Kosten. Dies bedeutet, dass ein Knoten zur Verfügung stehen muss, der die Speicherkapazität für mögliche Wege bereitstellt und die Rechenkapazität für den Suchalgorithmus beinhaltet.
- Zeitverhalten: Das Zeitverhalten – beispielsweise Klemme-Klemme-Reaktionszeit – ist für einmalig ausgeführte Funktionen – Weg suchen oder wählen sowie Reservierung – positiv, weil diese Funktionen ohne die Kommunikation über ein Netzwerk autark ablaufen. Für wiederholt ausgeführte Funktionen – Weg öffnen und schließen – ist das Zeitverhalten geringfügig schlechter, es ist jedoch immer noch positiv zu bewerten. Die Funktionen Weg öffnen und Weg schließen erfordern zwar eine Kommunikation über ein Netzwerk, da diese aber unmittelbar stattfindet (siehe Abbildung 8.2 auf Seite 89), ist kein Zeitverlust zu erwarten.

Nachteile: Dieses Entwurfsmuster hat verschiedene Nachteile, die im Folgenden aufgelistet werden:

- Verfügbarkeit: Fällt der Knoten aus, der für das Suchen eines Wegs zuständig ist, kann für die gesamte Anlage beziehungsweise Teilanlage kein Weg mehr gesucht werden. Anlagenstillstand ist die Folge.
- Ressourcennutzung: Es wird ein leistungsstarker Knoten benötigt, der die gesamte Speicherkapazität für mögliche Wege des Systems zur Verfügung stellen muss, alle Karten und Wege aktualisiert sowie die Rechenkapazität für einen Suchalgorithmus beinhaltet. In diesem Knoten müssen alle möglichen Wege verwaltet und gesucht werden.

8.3 Ableitung von UML-Modellen für die Anwendung der Entwurfsmuster

Nachdem im vorherigen Abschnitt jeweils ein mögliches *Funktions- und Verteilungsmuster* für die Anlagenfunktion *Transportieren* beschrieben wurde, wird im Folgenden das in Abschnitt 7.4 beschriebene UML-Modell für die unterschiedlichen Sichten auf die Entwurfsmuster am Beispiel der Anlagenfunktion *Transportieren* angewandt.

8.3.1 Funktionsmuster

Im Nachfolgenden wird das in Abschnitt 7.4 beschriebene UML-Modell auf das *Funktionsmuster* angewendet und am Beispiel der Anlagenfunktion *Transportieren* aufgezeigt. Wie zuvor beschrieben, erlaubt das UML-Modell unterschiedliche Sichten auf ein Entwurfsmuster. Diese einzelnen Sichten werden im Weiteren erläutert.

Programmsicht

Die *Programmsicht* der Funktionsmuster stellt den prinzipiellen und allgemeingültigen Aufbau der Funktionsmuster dar. Das Klassendiagramm in Abbildung 8.3 auf Seite 93 zeigt den allgemeinen Aufbau und die wesentlichen Eigenschaften des Funktionsmusters sowie dessen Beziehungen. Im Nachfolgenden werden die ontologischen Kernelemente des Klassendiagramms beschrieben.

Ein Funktionsmuster (Klasse *Muster*) besteht aus dessen allgemeinen Beschreibungsmerkmalen, wie beispielsweise Kontext, Name, Mustertyp, Teilnehmer, Musterkategorie und Zweck, sowie einer Lösung und einer Anlagenfunktion. Da eine Lösung (Klasse *Lösung*) ohne ein Muster nicht existieren kann, wurde diese Existenzbeziehung durch eine Komposition dargestellt. Ein Muster besitzt genau eine Lösung, und diese Lösung gehört auch nur zu genau einem Muster. Diese Beziehung wird in der Komposition durch die Multiplizität 1 ↔ 1 dargestellt. Ein Funktionsmuster beinhaltet genau eine Anlagenfunktion, die wiederum zu genau einem Muster gehört. Diese Beziehung wird in der Assoziation durch die Multiplizität 1..* ↔ 1 dargestellt. Eine Anlagenfunktion (Klasse *Anlagenfunktion*) besteht aus dem allgemeinen Attribut Name, das die Anlagenfunktion namentlich beschreibt.

Die Lösung (Klasse *Lösung*) des Entwurfsmusters besteht aus mindestens einer Automatisierungsfunktion (Klasse *Funktion*), die Bestandteil genau einer Lösung ist. Eine Automatisierungsfunktion (Klasse *Funktion*) besteht aus dem allgemeinen Attribut Name. Da eine Lösung immer durch Automatisierungsfunktionen realisiert wird und eine Automatisierungsfunktion ohne Lösung nicht existieren kann, wird dies mithilfe einer Komposition dargestellt. Die zuvor beschriebene Beziehung zwischen Lösung und Automatisierungsfunktion wird in der Komposition durch die Multiplizität 1 ↔ 1..* dargestellt.

Eine Automatisierungsfunktion (Abstrakte Klasse *Funktion*) wird untergliedert in *Funktion ohne Unterfunktionen* und *Funktion mit Unterfunktionen.* Dargestellt wird diese Untergliederung durch Vererbung. Eine *Funktion mit Unterfunktionen* besteht aus einer oder mehreren *Funktionen.* Dies wird mithilfe der Assoziation 1..* dargestellt.

Des Weiten kann die Klasse *Funktion ohne Unterfunktionen* aus einem oder keinem $XORS$et bestehen. Ein $XORS$et beinhaltet mindestens zwei *Funktionen ohne Unterfunktionen.* Diese Assoziation wird durch die Multiplizität 0..1 ↔ 2..* dargestellt. Die Klasse *Funktion mit Unterfunktionen* besteht aus keinem $XORS$et oder mehreren $XORS$ets, die wiederum zu genau einer *Funktion mit Unterfunktionen* gehören. Diese Assoziation wird durch die Multiplizität 0..* ↔ 1 dargestellt. Die Klasse $XORSet$ besteht aus den Attributen OR und XOR. Diese Attribute werden benötigt, um den Zusammenhang zwischen den Funktionen (OR- oder XOR-Beziehung zwischen den Funktionen) zu definieren. Beide Attribute können nicht gleichzeitig $TRUE$ oder $FALSE$ sein.

Eine Funktion kann aus keinen oder mehreren Verbindungen (Klasse *Verbindung*) zu anderen Automatisierungsfunktionen bestehen. Sollte eine Verbindung bestehen, gehört diese zu genau zwei Automatisierungsfunktionen. Diese Beziehung wird in der Assoziation durch die Multiplizität 2 ↔ 0..* dargestellt.

Eine Verbindung kann wiederum vom Typ Zyklische Verbindung (Klasse *Zyklische Verbindung*) oder Azyklische Verbindung (Klasse *Azyklische Verbindung*) sein. Dies wird im Klassendiagramm durch Vererbung dargestellt. Die nachfolgende Abbildung 8.3 zeigt die allgemeingültige Sicht eines Funktionsmusters in Form eines Klassendiagramms.

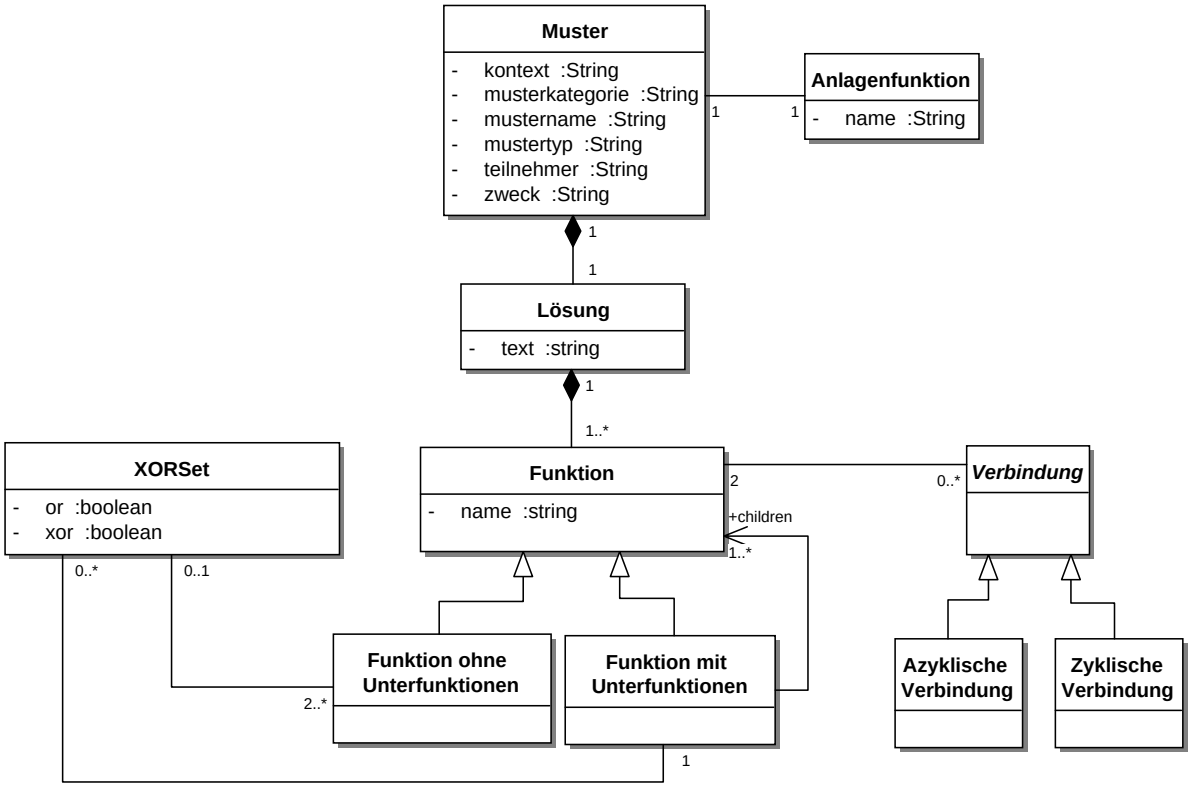

Abbildung 8.3: Programmsicht des Funktionsmusters

Mustersicht

Diese Sicht der Funktionsmuster ermöglicht eine musterbezogene Darstellung für jedes Funktionsmuster. Da in dieser Sicht kein Verhalten modelliert werden soll, muss für die Darstellung der spezifischen Sicht aus den zur Verfügung stehenden Strukturdiagrammen der UML das für diese Problemstellung optimale UML-Diagramm ausgewählt werden. Das Klassendiagramm stellt, wie zuvor erläutert, die statische Struktur eines Funktionsmusters dar [vgl. Rupp et al. 2007, S. 101] und ist somit für eine musterbezogene Darstellung am besten geeignet. Diese Darstellungsform ermöglicht eine Sicht auf jedes Funktionsmuster. In dieser Sicht beschreiben die einzelnen Klassen nicht den grundlegenden Aufbau eines Entwurfsmusters, sondern beziehen sich auf ein konkretes Funktionsmuster. Im Klassendiagramm (siehe Abbildung 8.4 auf Seite 95) wird das Muster »Transportieren mit Wegverfolgung und Reservierung« beschrieben, sowie auch seine Eigenschaften und Assoziationen.

Das Funktionsmuster »Transportieren mit Wegverfolgung und Reservierung« besteht aus einer Lösung, welche die konkreten Automatisierungsfunktionen *Weg suchen*, *Weg wählen*, *Weg reservieren*, *Weg öffnen*, *Weg schließen* und *Reservierung aufheben* beinhaltet. Die Eigenschaften

der einschlägigen Automatisierungsfunktion bestehen aus den jeweiligen Ausprägungen, wie beispielsweise »name=Weg suchen«. Die Automatisierungsfunktionen sind durch eine zyklische Verbindung miteinander verbunden, da unidirektional Informationen ausgetauscht werden müssen.

Die Funktion *Weg suchen* besteht aus den Unterfunktionen *Wegverfolgung* und *Wegfindungsalgorithmus*, die mithilfe einer zyklischen Verbindung miteinander verbunden sind. Das *XORSet* der Klasse *Wegverfolgung* legt fest, dass der Anwendungsentwickler zwischen der *Diskreten* oder *Kontinuierlichen Wegverfolgung* auswählen muss. Beim *XORSet* der Klasse *Weg wählen* kann zwischen *Default* – und *Alternativ Route* gewählt werden.

Da die Funktion *Weg öffnen* eine Auswahl von möglichen Unterfunktionen erlaubt, und der Anwendungsentwickler aus diesem Angebot beliebig viele Unterfunktionen auswählen kann, beinhalten die Klassen *Transportbänder anschalten* und *Ventile öffnen* ihre jeweiligen Ausprägungen, wie beispielsweise »name=Transportband anschalten«. Die Klasse *Weg öffnen* kann beliebig viele Instanzen der Klassen *Transportbänder anschalten* und *Ventile öffnen* beinhalten. Diese Assoziation wird durch die Multiplizität 1..* dargestellt. Abbildung 8.4 zeigt eine exemplarische Sicht in Form eines Klassendiagramms für das Funktionsmuster »Transportieren mit Wegverfolgung und Reservierung«.

Anwendungssicht

Die *Anwendungssicht* beschreibt eine konkrete Anwendung eines Entwurfsmusters und die dafür erstellten Objekte. Deshalb wird für die Darstellung dieser Sicht das Objektdiagramm verwendet, das eine konkrete Instanz des in der Mustersicht beschriebenen grundsätzlichen Aufbaus eines Funktionsmusters darstellt. Die nachfolgende Erläuterung zeigt für das Funktionsmuster »Transportieren mit Wegverfolgung und Reservierung« eine exemplarische Sicht in Form eines Objektdiagramms. In dieser Instanz werden die Automatisierungsfunktionen *Wegverfolgung*, *Weg wählen*, *Weg öffnen* und *Weg schließen* durch deren Unterfunktionen *Kontinuierliche Wegverfolung*, *Default Route*, *Transportbänder anschalten*, *Transportbänder ausschalten* realisiert. Deshalb sind die in der Mustersicht definierten »XORSets« und die Klassen *Diskrete Wegverfolgung*, *Alternative Route*, *Ventile öffnen* und *Ventile schließen* in der Anwendungssicht nicht mehr von Bedeutung. Abbildung 8.5 auf Seite 96 zeigt eine konkrete Instanz des Funktionsmusters »Transportieren mit Wegverfolgung und Reservierung« in Form eines Objektdiagramms.

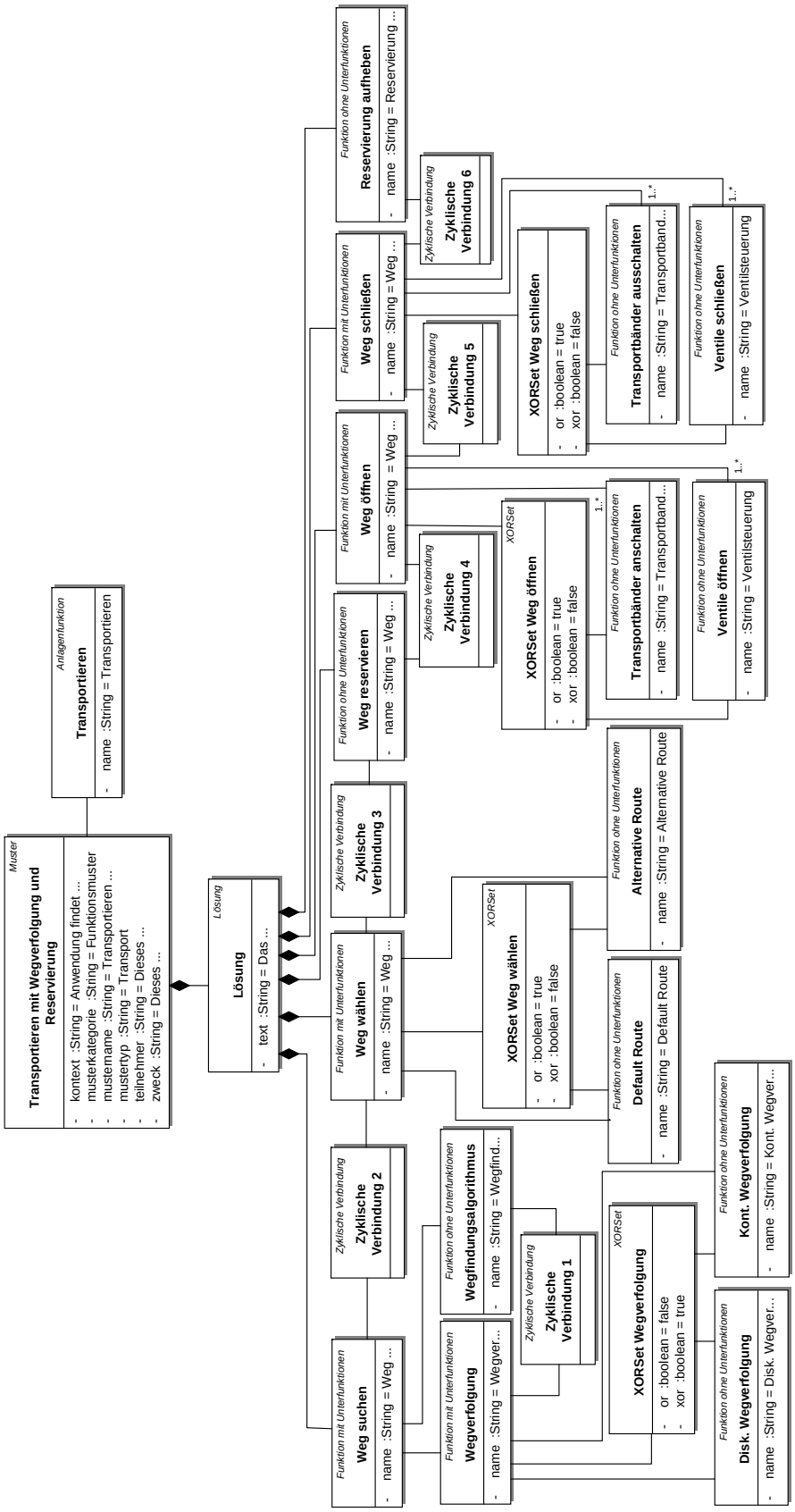

Abbildung 8.4: Mustersicht des Funktionsmusters

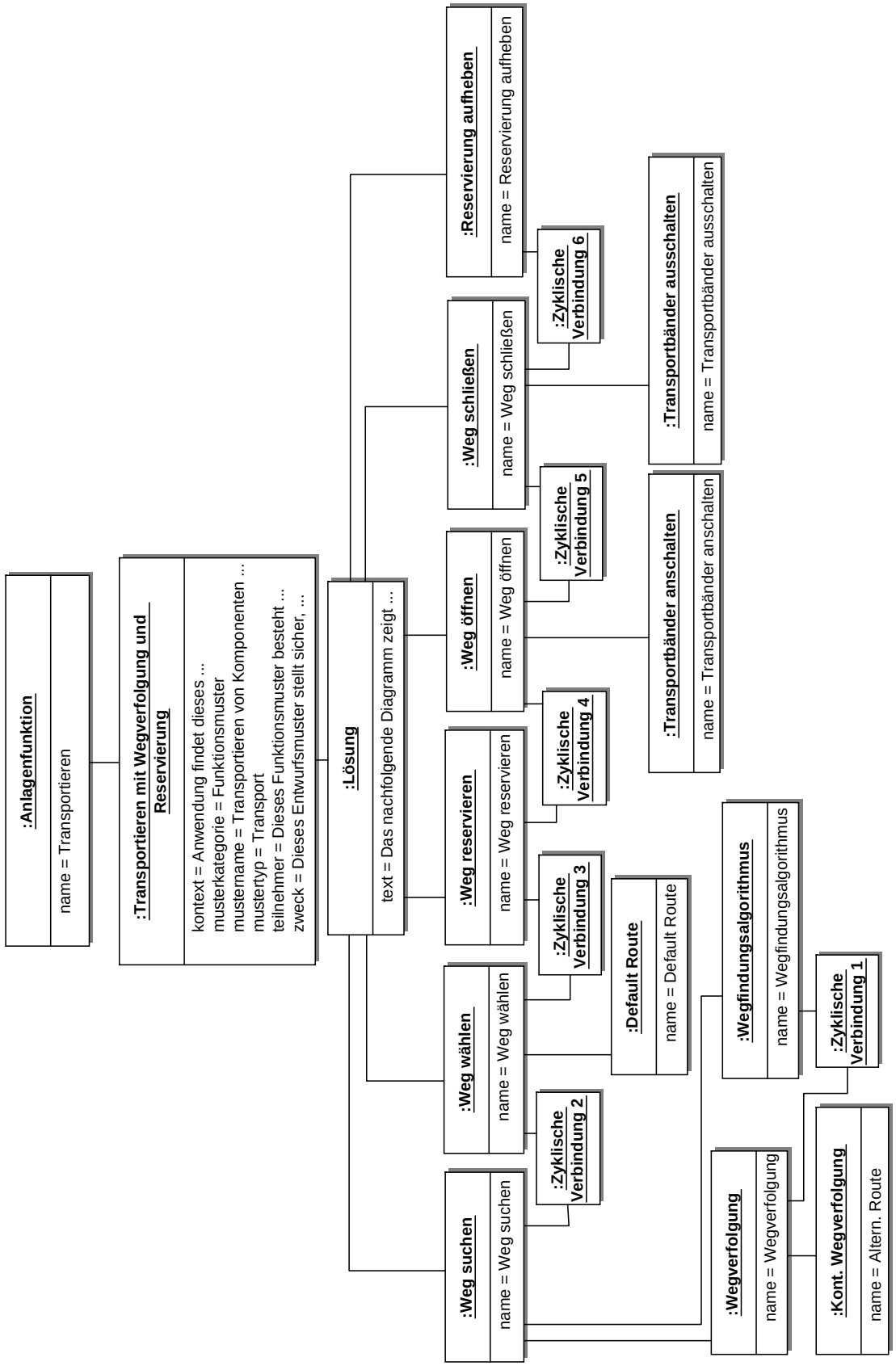

Abbildung 8.5: Anwendungssicht des Funktionsmusters

8.3.2 Verteilungsmuster

Im Weiteren wird das in Abschnitt 7.4 beschriebene UML-Modell auf das *Verteilungsmuster* angewendet und am Beispiel einer Verteilungsalternative für die Funktionalität der Anlagenfunktion *Transportieren* aufgezeigt. Wie zuvor beschrieben, erlaubt das UML-Modell unterschiedliche Sichten auf ein Entwurfsmuster. Diese einzelnen Sichten werden im Nachfolgenden erläutert.

Programmsicht

Die *Programmsicht* der Verteilungsmuster stellt den prinzipiellen und allgemeingültigen Aufbau der Verteilungsmuster dar. Das Klassendiagramm in Abbildung 8.6 auf der nächsten Seite zeigt den allgemeinen Aufbau und die wesentlichen Eigenschaften des Verteilungsmusters sowie die Beziehungen. Im Folgenden werden die ontologischen Kernelemente des Klassendiagramms beschrieben.

Ein Verteilungsmuster (Klasse *Muster*) besteht aus dessen allgemeinen Beschreibungsmerkmalen, wie beispielsweise Kontext, Musterkategorie, Mustername, Mustertyp, Nachteile, Vorteile, Teilnehmer, Zweck sowie einer Lösung (Klasse *Lösung*). Da eine Lösung ohne ein Muster nicht existieren kann, wurde diese Existenzbeziehung durch eine Komposition dargestellt. Ein Muster besitzt genau eine Lösung, und diese Lösung gehört auch nur zu genau einem Muster. Diese Beziehung wird in der Komposition durch die Multiplizität $1 \leftrightarrow 1$ dargestellt. Die Lösung des Entwurfsmusters besteht aus mindestens einer Hardware (Knoten, Sensor und Aktor), die wiederum zu einer Lösung gehört. Diese Beziehung wird in der Assoziation durch die Multiplizität $1..* \leftrightarrow 1$ dargestellt. Des Weiteren besteht die Lösung aus mindestens einem Funktionsblock, der wiederum zu einer Lösung gehört. Diese Beziehung wird in der Assoziation durch die Multiplizität $1..* \leftrightarrow 1$ dargestellt. Dieser Funktionsblock kann auf eine Hardware verteilt werden. Auf ihr kann kein oder können mehrere Funktionsblöcke implementiert werden. Deshalb wird diese Assoziation zwischen Hardware und Funktionsblock durch die Multiplizität $1 \leftrightarrow 0..*$ dargestellt.

Die Klasse *Hardware* sowie die Klasse *Funktionsblock* erben beide von der Klasse *Portträger*. Dadurch kann garantiert werden, dass ein Port (Klasse *Eingangsport* oder Klasse *Ausgangsport*) zu genau einer Hardware oder Funktion gehört und eine Hardware oder Funktion aus keinen oder mehreren Ports besteht. Deshalb wird diese Assoziation zwischen Portträger und Port durch die Multiplizität $1 \leftrightarrow 0..*$ dargestellt. Sobald zwei Ports miteinander verbunden werden, entsteht auch eine Verbindung (Klasse *Steuerverbindung* oder Klasse *Datenverbindung*). Ein Port besitzt genau eine Verbindung, und eine Verbindung gehört zu genau zwei Ports. Deshalb wird diese Assoziation zwischen Port und Verbindung durch die Multiplizität $2 \leftrightarrow 1$ dargestellt. Die jeweiligen Anforderungs- und Lösungsmerkmale wurden den Merkmalsträgern als Eigenschaft hinzugefügt, wie beispielsweise die Eigenschaft »bereitgestellter Speicher« in der Klasse *Hardware*. Die nachfolgende Abbildung 8.6 zeigt die allgemeingültige Sicht eines Funktionsmusters in Form eines Klassendiagramms.

Mustersicht

Diese Sicht der Verteilungsmuster soll eine musterbezogene Darstellung für jedes Verteilungsmuster ermöglichen. Da in dieser Sicht kein Verhalten modelliert werden soll, muss für die Darstellung der spezifischen Sicht aus den zur Verfügung stehenden Strukturdiagrammen der UML das für diese Problemstellung optimale UML-Diagramm ausgewählt werden. Das Klas-

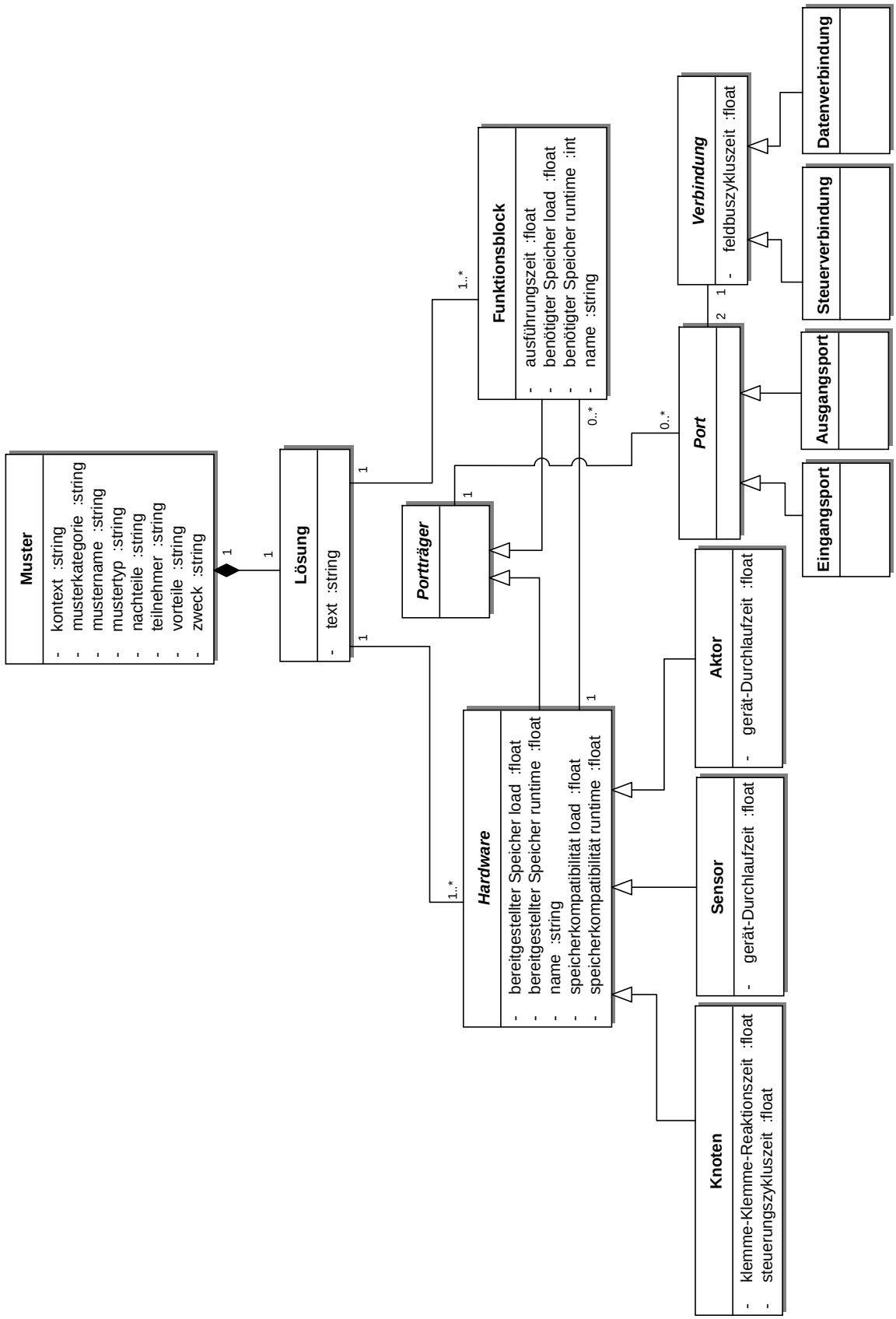

Abbildung 8.6: Programmsicht des Verteilungsmusters

sendiagramm stellt, wie zuvor erläutert, die statische Struktur eines Verteilungsmusters dar [vgl. Rupp et al. 2007, S. 101] und ist für eine musterbezogene Darstellung anwendbar. Die Darstellung von Ergänzungen, wie beispielsweise zwei Transportwege anstatt einem, ist in einem »reinen« Klassendiagramm schwer darstellbar.

Das Paketdiagramm wird verwendet, um unterschiedliche Sichten auf ein System und dessen Strukturierung zu ermöglichen [vgl. Rupp et al. 2007, S. 165]. Da das Paketdiagramm in seiner Notation eine Einbindung von Klassen erlaubt, wird dieses Diagramm verwendet. Die Darstellung von Ergänzungen wird dadurch erleichtert.

In dieser Sicht beschreiben die einzelnen Klassen nicht den grundlegenden Aufbau eines Entwurfsmusters, sondern beziehen sich auf ein konkretes Verteilungsmuster. In den nachfolgenden Paketdiagrammen wird das Muster »Verteiltes Transportieren inklusive Wegverfolgung und Reservierung« beschrieben sowie dessen Eigenschaften und Assoziationen. Das Verteilungsmuster besteht aus zwei grundlegenden Paketen:

1. Lösungskern
2. Lösungsergänzung (zum Beispiel bei einem zusätzlichen Transportweg)

In diesen Paketen ist die statische Struktur der jeweiligen Klassen eines Verteilungsmusters dargestellt. Die nachfolgenden Abbildungen (siehe Abbildung 8.7 auf der nächsten Seite und Abbildung 8.8 auf Seite 101) zeigen beide eine exemplarische Sicht in Form eines Paketdiagramms für das Verteilungsmuster »Verteiltes Transportieren inklusive Wegverfolgung und Reservierung«. Zwischen den Paketen des Verteilungsmusters existieren Abhängigkeiten, die im Diagramm aus Gründen der Übersichtlichkeit nicht dargestellt wurden. Diese Abhängigkeiten sind notwendig, wenn ein zusätzlicher Transportweg benötigt wird. Dies sind:

1. Die Klasse »Eingangsport 18« (Lösungsergänzung) wird mit der Klasse »Knoten 1« (Lösungskern) verbunden.
2. Die Klasse »Eingangsport 19« (Lösungsergänzung) wird mit der Klasse »Reservierung aufheben« (Lösungskern) verbunden.
3. Die Klasse »Eingangsport 20« (Lösungsergänzung) wird mit der Klasse »Knoten 1« (Lösungskern) verbunden.
4. Die Klasse »Eingangsport 21« (Lösungsergänzung) wird mit der Klasse »Wegverfolgung« (Lösungskern) verbunden.
5. Die Klasse »Ausgangsport 12« (Lösungsergänzung) wird mit der Klasse »Weg reservieren« (Lösungskern) verbunden.
6. Die Klasse »Ausgangsport 13« (Lösungsergänzung) wird mit der Klasse »Knoten 1« (Lösungskern) verbunden.
7. Die Klassen der Lösungsergänzung vom Typ »Funktionsblock, Aktor, Sensor und Knoten« werden mit der Klasse »Lösung« des Lösungskerns verbunden.

Anwendungssicht

Die Anwendungssicht beschreibt eine konkrete Anwendung eines Entwurfsmusters und die dafür erstellten Objekte. Deshalb wird für die Darstellung dieser Sicht das Objektdiagramm verwendet, das eine konkrete Instanz des in der Mustersicht beschriebenen grundsätzlichen Aufbaus eines Verteilungsmusters darstellt. Abbildung 8.9 auf Seite 102 zeigt für das Verteilungsmuster

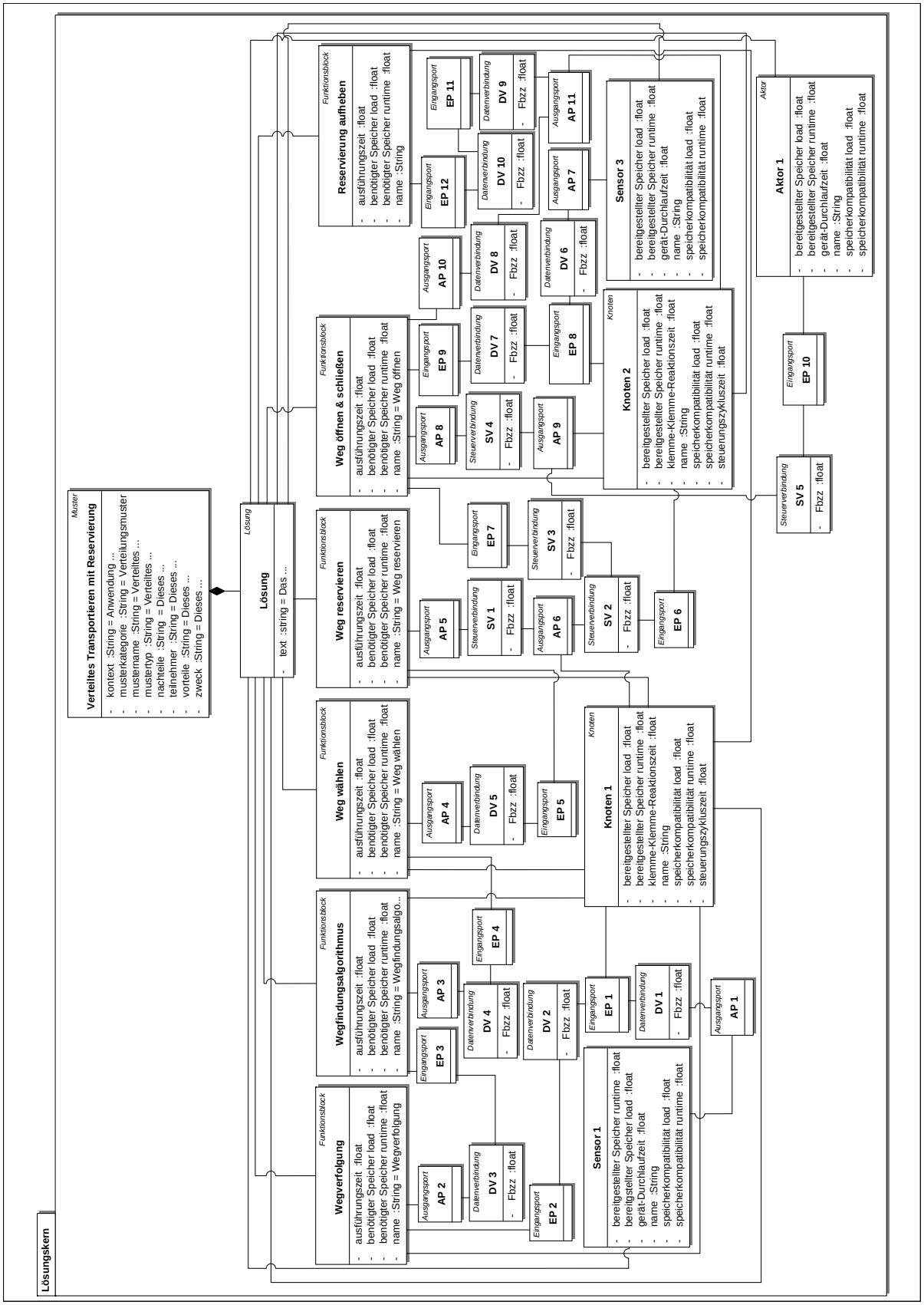

Abbildung 8.7: Mustersicht des Verteilungsmusters – Lösungskern

AP → Ausgangsport EP → Eingangsport DV → Datenverbindung SV → Steuerverbindung Fbzz → Feldbuszykluszeit

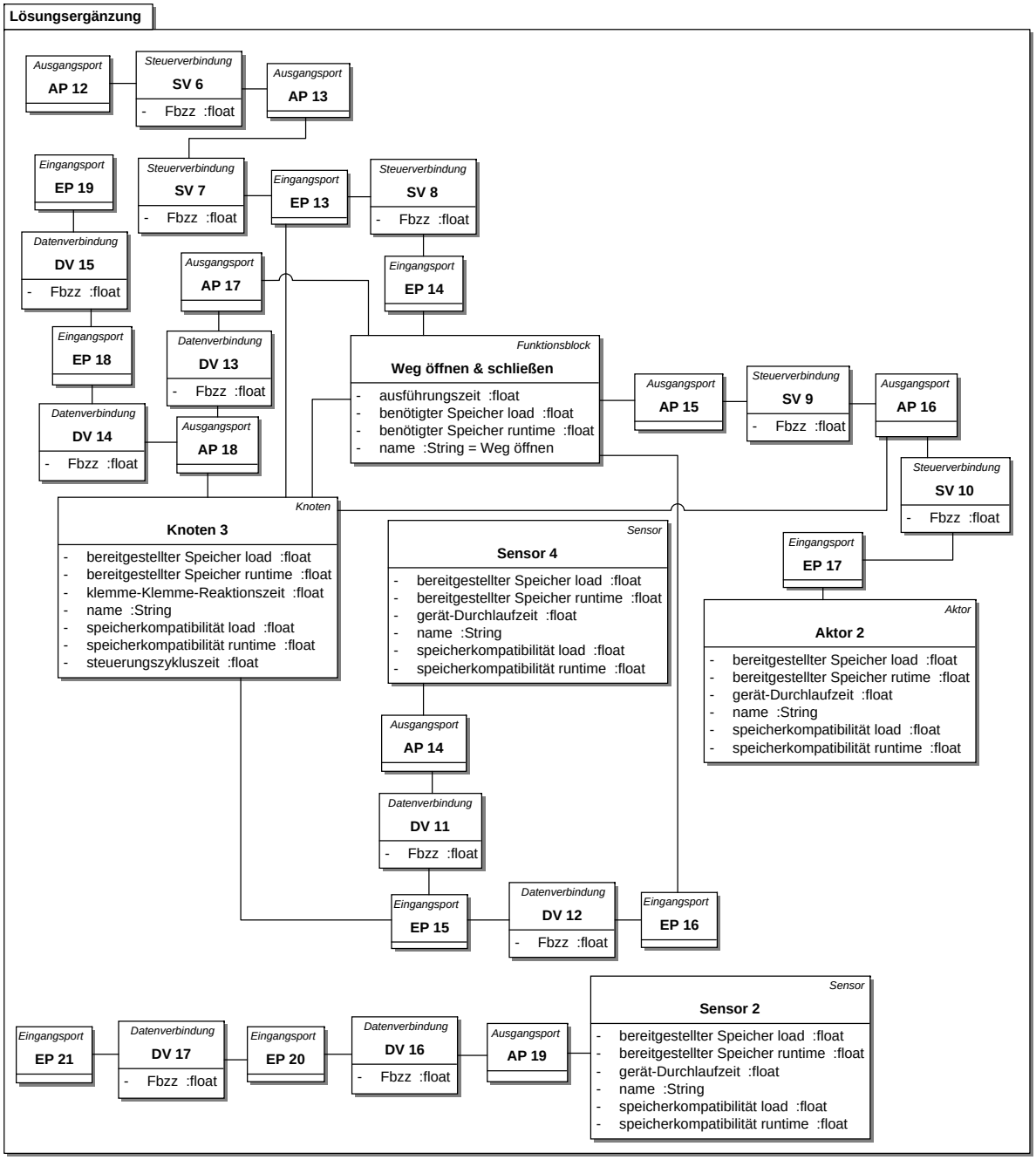

Abbildung 8.8: Mustersicht des Verteilungsmusters – Lösungsergänzung

AP → Ausgangsport EP → Eingangsport DV → Datenverbindung SV → Steuerverbindung
Fbzz → Feldbuszykluszeit

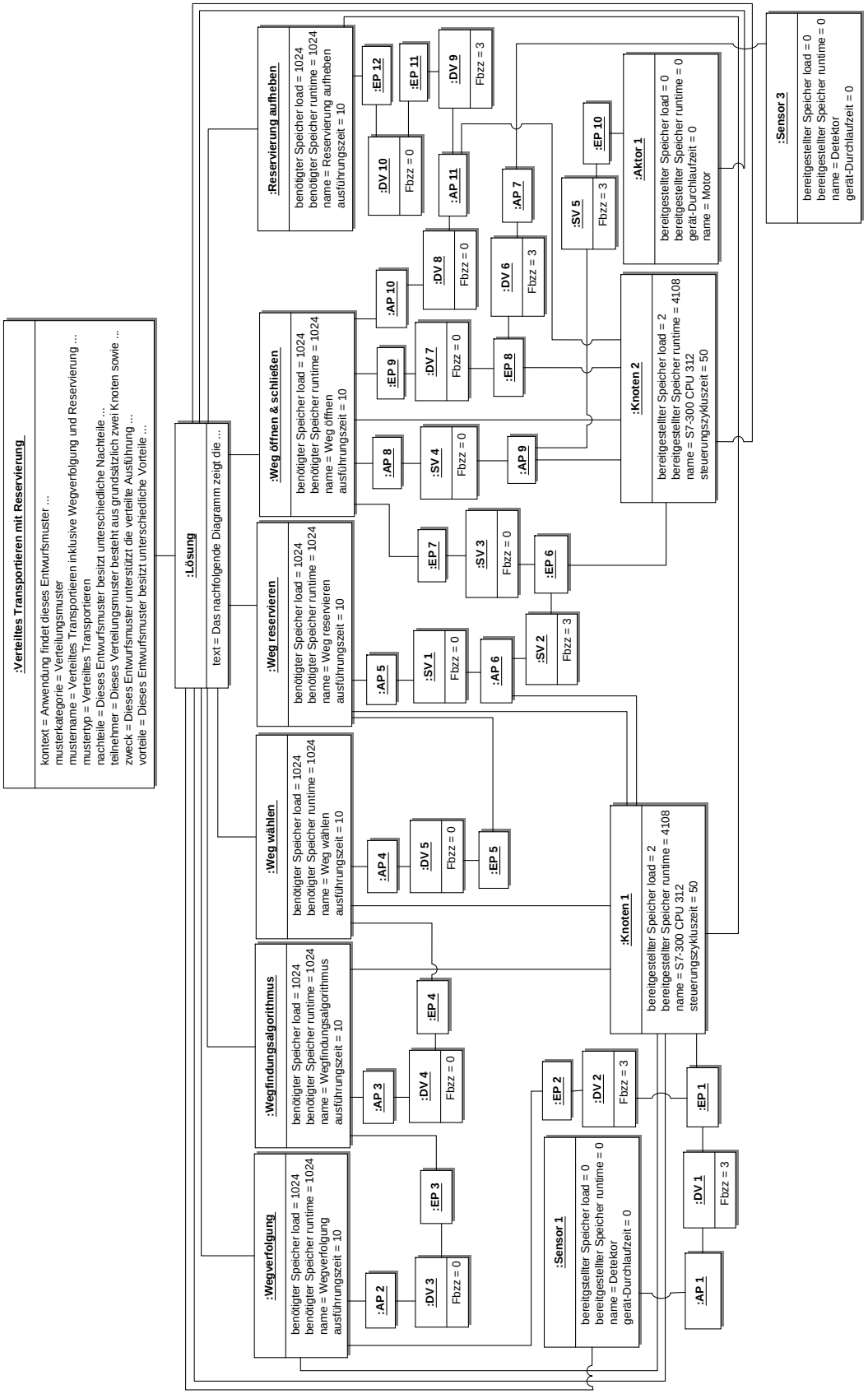

Abbildung 8.9: Anwendungssicht des Verteilungsmusters

AP → Ausgangsport EP → Eingangsport DV → Datenverbindung SV → Steuerverbindung Fbzz → Feldbuszykluszeit

»Verteiltes Transportieren inklusive Wegverfolgung und Reservierung« eine exemplarische Sicht in Form eines Objektdiagramms. Alle Funktionsblöcke der Automatisierungsfunktionen *Weg suchen* sowie *Weg öffnen & schließen* werden auf die im Diagramm beschriebene Hardware verteilt.

9 Beispielhafte Anwendung des Konzepts

Die Eignung des zuvor beschriebenen Konzepts, den Entwurfsprozess zielgerichtet zu leiten und zu unterstützen, wurde erforscht. Hierzu wird im Nachfolgenden das entwickelte Konzept hinsichtlich praktischer Eignung anhand von zwei Anwendungsbeispielen (fertigungstechnische und verfahrenstechnische Anlage) aufgezeigt. Dadurch soll der Nachweis erbracht werden, dass das entwickelte Konzept den Anwendungsentwickler zielgerichtet beim Entwurf verteilter Automatisierungssysteme unterstützt. Die für die Anwendungsbeispiele erstellten Modelle sowie deren Erläuterungen befinden sich im Anhang E und Anhang F.

Basierend auf der Einordnung der nfAs in das Vorgehensmodell, beschrieben in Abschnitt 6.2.1, erfolgt im nachfolgenden Beispiel eine Betrachtung der jeweils relevanten nfAs. Die nfAs *Wiederverwendbarkeit* und *Testbarkeit* wurden der *Ebene 4* zugeordnet. Die nfA *Testbarkeit* wird erst dann relevant, wenn der rechte Ast des V-Modells durchlaufen wird, was nicht Gegenstand des Forschungsprojekts FAVA und somit nicht Bestandteil dieser Arbeit ist. Die nfAs *Ressourcennutzung*, *Zeitverhalten* und *Installierbarkeit* wurden der *Ebene 2* zugeordnet. Die nfA *Installierbarkeit* kann erst überprüft werden, wenn die Installation der Software erfolgt ist. Ein Ziel des Forschungsprojekts FAVA ist es, den Steuerungsentwurf verteilter Systeme zu vereinfachen und nicht deren Implementierung. Deshalb ist die nfA *Installierbarkeit* nicht Bestandteil dieser Arbeit. Da in den Verteilungsmustern eine Berücksichtigung der nfAs stattfindet und diese auf *Ebene 2* die Verteilung der Funktionalität unterstützen, erfolgt im nachfolgenden Beispiel eine Betrachtung der nfAs *Zeitverhalten* und *Ressourcennutzung*.

9.1 Fördertechnische Anlage

Im Nachfolgenden wird die Anwendbarkeit des zuvor beschriebenen Konzepts anhand einer fertigungstechnischen Anlage aufgezeigt. Hierzu erfolgt zum einen eine allgemeine Beschreibung der Anlage, welche die Anforderungen an die Anlage beinhaltet, und zum anderen die Anwendung des Konzepts der Entwurfsmuster.

9.1.1 Beschreibung der Anlage

Die im Kontext dieses Kapitels untersuchte *Metalltrennungsanlage* aus dem Bereich der Fertigungstechnik dient der automatisierten Trennung von metallischem und nicht-metallischem Material eines kontinuierlichen Schüttgutstroms und stellt somit ein in der praktischen Anwendung typisches Beispiel dar (siehe Abbildung 9.1 auf Seite 106). Die *Metalltrennungsanlage* besteht aus einem Anlagenteil mit drei Gurtförderern. Über einen induktiven Sensor (Detektor) werden die Metallstücke im Förderstrom des zuführenden Gurtförderers detektiert und am Abwurf des zuführenden Gurtförderers durch Luftdruck (Düse) auf den Gurtförderer 3 aussortiert. Durch diesen Gurtförderer werden die Metallstücke in einen Container transportiert, dessen Füllstand überwacht und regelmäßig gespeichert werden soll. Die nicht-metallischen Stücke werden über den Gurtförderer 2 weitertransportiert.

Der Anlagenbetreiber stellt an die zuvor beschriebene *Metalltrennungsanlage* eine Vielzahl von funktionalen und nicht-funktionalen Anforderungen. Das Gesamtsystem soll modular aufgebaut sein, sodass sinnvolle Module als Ganzes ausgetauscht werden können. Des Weiteren sollen insgesamt vier Steuerungen im System verbaut werden. Sobald der Motor eines Bandes eine zu hohe Betriebstemperatur erreicht, müssen alle Bandmotoren stoppen. Der Füllstand des Containers wird jede Minute gemessen und für 10 Jahre archiviert (float: 4 Byte), wobei ein voller Container einen Stopp aller Bänder innerhalb von 0,5 s zur Folge hat. Der Abstand zwischen Detektor und Düse beträgt 20 cm. Die Anzahl der von der Düse aussortierten Metallstücke soll einmal stündlich gespeichert und über 10 Jahre archiviert werden (max. Teile pro Stunde: 10 000; int: 4 Byte). Bei einem Not-Halt der Anlage ertönt an allen drei Hupen ein Signal (Signalton 0,5 s an 2 s aus).

Die zuvor dargestellte Prozess- und Anforderungsbeschreibung an die Automatisierung der *Metalltrennungsanlage* stellt eine funktionsorientierte Perspektive für das spätere Sollverhalten der Anlage dar. In Bezug auf das Engineering von Automatisierungsanlagen lässt sich diese Perspektive in die Planungsphase des Engineerings einordnen. In dieser Phase wird das Lastenheft erstellt, das die Prozess- und Anforderungsbeschreibung der *Metalltrennungsanlage* enthält.

Der Anwendungsentwickler muss in den nachfolgenden Engineering-Phasen diese Anforderungen analysieren und die Steuerungsarchitektur entwerfen, die in späteren Engineering-Phasen implementiert wird. Für die Erstellung des Steuerungsentwurfs kann der Anwendungsentwickler zusätzlich zur textuellen und zumeist informellen Spezifikation des Prozesses auf eine Übersicht der in der Anlage verbauten Sensoren, Aktoren und Steuerungen zurückgreifen (siehe Geräte- und SPS-Datenblatt in Anhang B – Tabelle B.1 und B.2 auf Seite 129). Die Prozess- und Anforderungsbeschreibung sowie das Geräte- und SPS-Datenblatt sind für eine Erstellung eines konzeptionellen Modells, welches das in den vorherigen Kapiteln beschriebene Konzept beinhaltet, notwendig. Die nachfolgende Abbildung 9.1 zeigt den zuvor beschriebenen schematischen Aufbau der *Metalltrennungsanlage*, die grundsätzlich aus drei Gurtförderern (inklusive Motor), drei Hupen, einem Detektor, einer Düse und einem Container (inklusive Füllstandsensor) besteht.

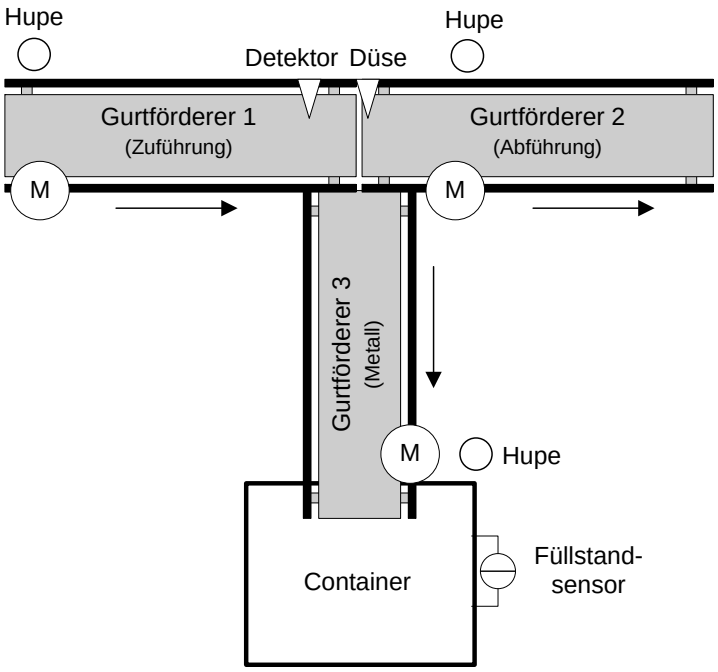

Abbildung 9.1: Schematischer Aufbau der Metalltrennungsanlage

9.1.2 Spezifikation des Entwurfs

Anhand der in Abschnitt 9.1.1 beschriebenen Prozess- und Anforderungsbeschreibung erfolgt im Nachfolgenden die Erstellung eines konzeptionellen Modells für die gesteuerte *Metalltrennungsanlage*. Dieses Modell beinhaltet – unter Berücksichtigung der gegebenen Anforderungen – alle beschriebenen Funktionen und deren Zusammenhänge mit notwendiger zu modellierender Steuerungshardware, Sensoren und Aktoren. Im Anwendungsfall der *Metalltrennungsanlage* wird im Folgenden aufgezeigt, wie der Entwurf verteilter Automatisierungssysteme, basierend auf der in Kapitel 6 beschriebenen Vorgehensweise und der in Kapitel 7 vorgestellten Entwurfsunterstützung, erfolgen kann.

Ebene 4

Auf Basis der in Abschnitt 6.1.2 beschriebenen Aktivitäten auf *Ebene 4* werden auf Grundlage der in Abschnitt 9.1.1 beschriebenen Anforderungen und des beschriebenen Anlagenlayouts die Ergebnisse der *Ebene 4* erstellt. Im Zuge der Automatisierung der *Metalltrennungsanlage* müssen verschiedene Aspekte berücksichtigt werden. Zum einen wird auf dieser Ebene das für die *Metalltrennungsanlage* notwendige *Komponentenmodell* erstellt, das die notwendigen Automatisierungsgeräte (zum Beispiel Sensoren und Aktoren) beinhaltet, und zum anderen das *Funktionsmodell*, das die funktionalen Anforderungen untergliedert.

In einem der nachfolgenden Schritte wird das *Komponentenmodell* erstellt (siehe Anhang E, Abschnitt E.1 sowie Abbildung E.1 auf Seite 168), das alle für die *Metalltrennungsanlage* relevanten Automatisierungsgeräte, wie beispielsweise Motor, Hupe, Temperatursensor, Füllstandsensor, Detektor und Düse, beinhaltet. Basierend auf den zuvor ermittelten funktionalen Anforderungen, können aus einer Vielzahl von Entwurfsmustern die für die aktuelle Problemstellung relevanten Entwurfsmuster (siehe Anhang C) ausgewählt und angewendet werden.

Dadurch kann der Anwendungsentwickler bei der Spezifizierung von funktionalen Anforderungen durch automatisierungstechnische Grundfunktionen – Automatisierungsfunktionen – unterstützt werden. Durch die Anwendung dieser *Funktionsmuster* wird das *Funktionsmodell* der *Metalltrennungsanlage* erstellt (siehe Anhang E, Abschnitt E.2 sowie Abbildung E.2 auf Seite 169).

Diese Automatisierungsfunktionen werden in späteren Entwurfsphasen durch Funktionsblöcke realisiert und auf die Hardware verteilt. Die Funktionsmuster beinhalten solche Vorschläge für Automatisierungsfunktionen und sind somit Grundlage für die Erstellung des *Funktionsmodells.* Grundsätzlich besteht die *Metalltrennungsanlage* aus den Anlagenfunktionen *Transportieren*, *Not-Halt*, *Temperatur überwachen*, *Sortieren*, *Hupen* und *Füllstand protokollieren.*

Exemplarisch werden im weiteren Verlauf die Aktivitäten auf den Ebenen anhand der Anlagenfunktion *Sortieren* der *Metalltrennungsanlage* dargestellt. Die vollständigen Ergebnisse im Hinblick auf die *Metalltrennungsanlage* können den Modellen in Anhang E, Abschnitt E.1 und E.2, entnommen werden. Abbildung 9.2 auf der nächsten Seite zeigt die Anwendung des *Funktionsmusters* am Beispiel der Anlagenfunktion *Sortieren.* Das *Funktionsmodell* besteht grundsätzlich aus den gleichen Automatisierungsfunktionen wie das Entwurfsmuster. Die speziellen »XOR«- oder »OR«-Funktionen des Funktionsmusters werden bei dessen Anwendung näher spezifiziert. Am Beispiel des Funktionsmusters *Sortieren* bedeutet dies, dass die im *Funktionsmuster* mögliche Auswahl von »XOR«-Funktionen bei der Anwendung des Entwurfsmusters durch die Automatisierungsfunktion *Düsensteuerung* näher spezifiziert wurde. Das Funktionsmuster *Sortieren* kann abhängig von der Anzahl an Sortierungen mehrmals Anwendung finden.

Ebene 3

Auf Basis der Ergebnisse von *Ebene 4* werden die Aktivitäten auf *Ebene 3* durchgeführt und diese Modelle somit näher spezifiziert (siehe Abbildung 9.3 auf Seite 109). Im Zuge der Automatisierung der *Metalltrennungsanlage* müssen verschiedene Aspekte berücksichtigt werden. Zum einen wird auf dieser Ebene das für die *Metalltrennungsanlage* notwendige *Topologiemodell* erstellt, das die konkreten Automatisierungsgeräte beschreibt, und zum anderen das *Softwaremodell*, das die Spezifizierung der Automatisierungsfunktionen durch ein oder mehrere Funktionsblöcke beinhaltet. Außerdem wird im *Topologiemodell* die Anzahl an Knoten festgelegt und die Knoten werden spezifiziert. Des Weiteren werden auf dieser Ebene die Funktionsblöcke mit der Automatisierungshardware, wie beispielsweise Sensoren und Aktoren, verknüpft. Die Ein- und Ausgänge der Automatisierungsgeräte der *Metalltrennungsanlage* können den Datenblättern in Anhang B entnommen werden. Die erstellten Modelle *Topologie- und Softwaremodell* der *Metalltrennungsanlage* sind in Anhang E, Abschnitt E.3 sowie Abbildung E.3 auf Seite 171 dargestellt.

Die nachfolgende Abbildung 9.3 zeigt exemplarisch die Aktivitäten und Ergebnisse auf *Ebene 3* am Beispiel der Anlagenfunktion *Sortieren.* In den Aktivitäten auf *Ebene 3* werden der Detektor, die Düse und die Knoten unter anderem durch die in den Datenblättern in Anhang B enthaltenen Daten bezüglich Ressourcennutzung, Ein- beziehungsweise Ausgänge sowie des Zeitverhaltens näher beschrieben. Die jeweiligen Ein- und Ausgänge werden im *Topologie- und Softwaremodell* als Eingangs- oder Ausgangsport dargestellt. Diese Merkmale in Bezug auf Ressourcennutzung und Zeitverhalten sind somit Lösungsmerkmale auf *Ebene 3* (siehe Tabelle 9.1 auf Seite 110), da sie auf dieser Ebene erarbeitet wurden. Tabelle 9.1 zeigt die Merkmale der Geräte, der Knoten und der Funktionen. Anhand dieser Merkmale, die in den nachfolgenden Ebenen als Anforderungsmerkmale dienen, kann auf *Ebene 2* die Verteilung der Funktionalität erfolgen.

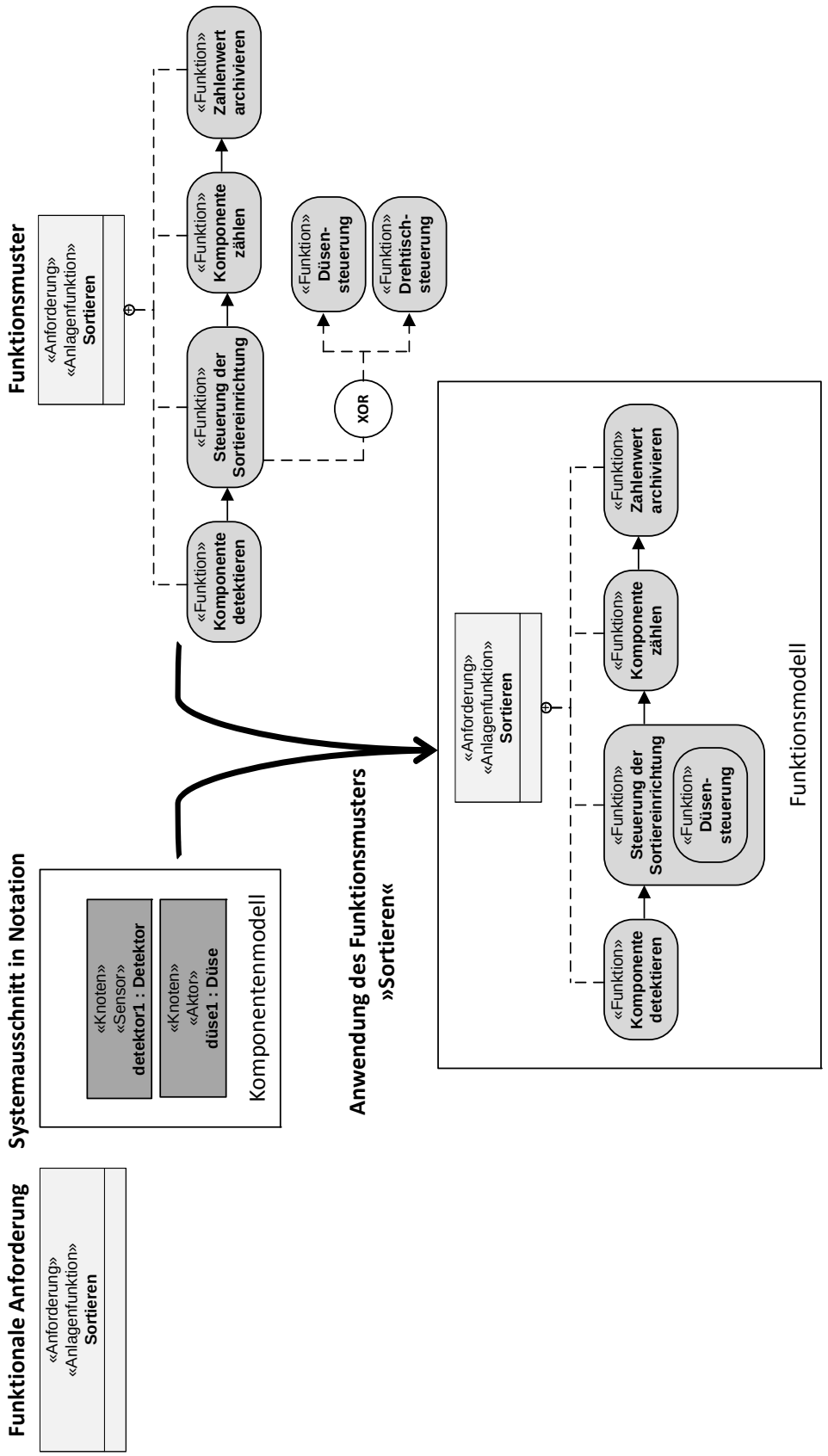

Abbildung 9.2: Anwendung eines Funktionsmusters

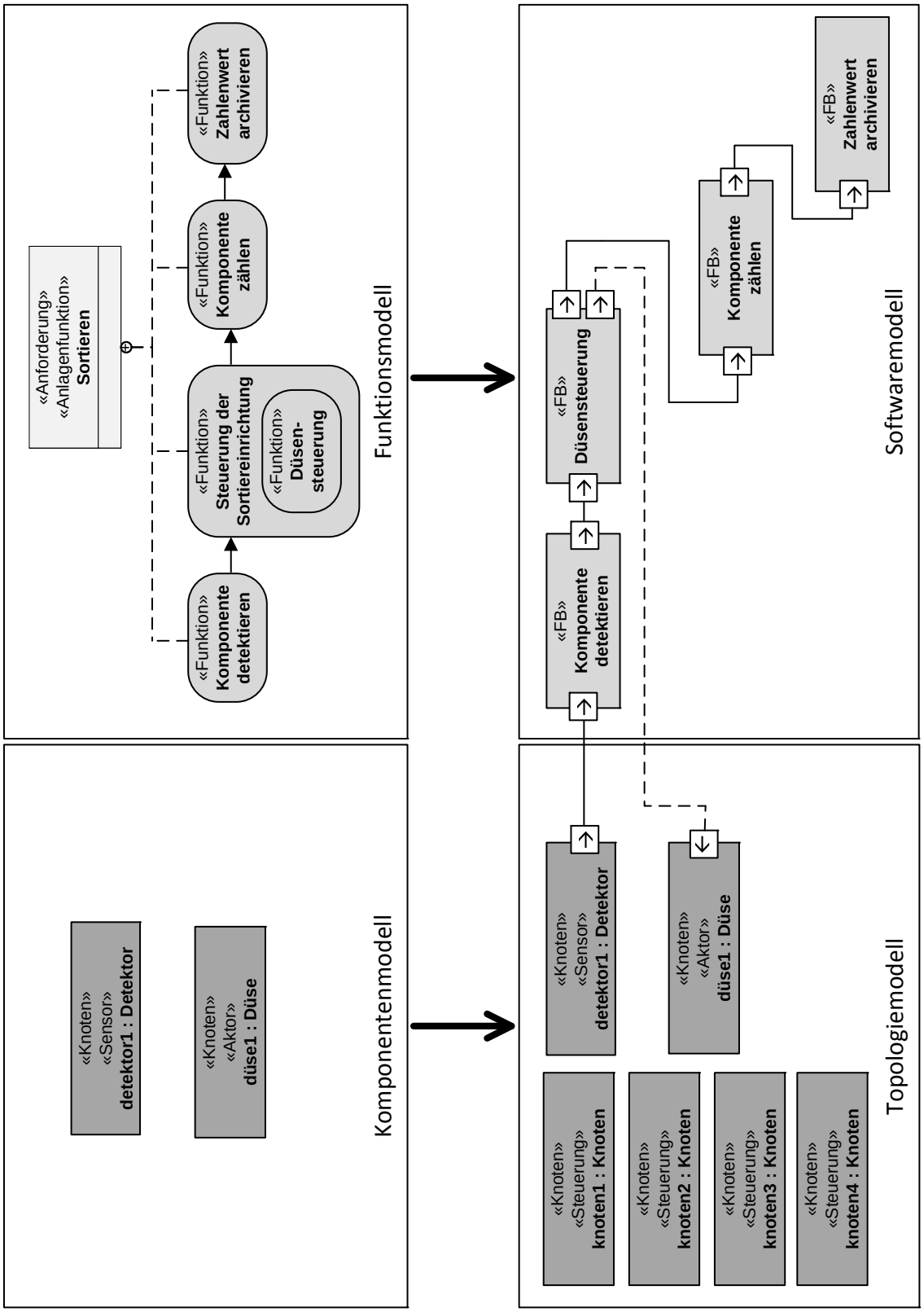

Abbildung 9.3: Aktivität auf Ebene 3 am Beispiel der Anlagenfunktion *Sortieren*

Tabelle 9.1: Lösungsmerkmale auf Ebene 3 (Ressourcennutzung und Zeitverhalten)

Lösungsmerkmale auf Ebene 3	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten 1	50 ms
Knoten 2	100 ms
Knoten 3	100 ms
Knoten 4	100 ms
Gerät-Durchlaufzeit	
Detektor	3 ms
Düse	3 ms
Ausführungszeit einer POU	
Komponente detektieren	10 ms
Düsensteuerung	10 ms
Komponente zählen	10 ms
Zahlenwert archivieren	10 ms
Feldbuszykluszeit	
Zykluszeit	30 ms
Bereitgestellter Speicher (Runtime)	
Detektor	10 KByte
Düse	–
Knoten 1	15 KByte
Knoten 2	19 KByte
Knoten 3	19 KByte
Knoten 4	19 KByte
Bereitgestellter Speicher (Load)	
Detektor	10 KByte
Düse	–
Knoten 1	2 MByte
Knoten 2	8 MByte
Knoten 3	8 MByte
Knoten 4	8 MByte
Benötigter Speicher (Runtime)	
Komponente detektieren	1024 Byte
Düsensteuerung	1024 Byte
Komponente zählen	1024 Byte
Zahlenwert archivieren	1024 Byte
Benötigter Speicher (Load)	
Komponente detektieren	1024 Byte
Düsensteuerung	1024 Byte
Komponente zählen	1024 Byte
Zahlenwert archivieren	1024 Byte

Ebene 2

Auf Basis der Ergebnisse von *Ebene 3* wird die Aktivität auf *Ebene 2* durchgeführt und das *Deploymentmodell* somit näher spezifiziert. Im Zuge der Automatisierung der *Metalltrennungsanlage* müssen verschiedene Aspekte berücksichtigt werden. Zum einen erfolgt die Verteilung der Funktionalität auf die Hardware und zum anderen muss diese Verteilung so stattfinden, dass die nfAs Zeitverhalten und Ressourcennutzung erfüllt werden. Die Verteilung der Funktionalität auf die Hardware wird im *Deploymentmodell* festgehalten. Basierend auf den zuvor ermittelten Automatisierungsfunktionen, können aus einer Vielzahl von Entwurfsmustern die Muster (siehe

Anhang C) ausgewählt und angewendet werden, die eine Verteilung der Funktionalität so ermöglichen, dass die nfAs erfüllt werden. Dadurch kann der Anwendungsentwickler bei der Verteilung und der Erfüllung von nfAs unterstützt werden. Durch die Anwendung dieser *Verteilungsmuster* wird das *Deploymentmodell* der *Metalltrennungsanlage* erstellt (siehe Anhang E, Abschnitt E.4 sowie Abbildung E.4 auf Seite 172). Abbildung 9.4 auf der nächsten Seite zeigt somit die Anwendung des *Verteilungsmusters* am Beispiel *Sortieren*. In der Aktivität auf *Ebene 2* werden die Lösungsmerkmale von *Ebene 3* zu Anforderungsmerkmalen, mithilfe derer die Lösungsmerkmale von *Ebene 2* berechnet werden können. Hierzu ist die Verteilung der Funktionalität zu berücksichtigen. Die Funktionalität der Komponente detektieren wird auf dem Detektor implementiert, die Funktionalitäten der Düsensteuerung, Komponente zählen und Zahlenwert archivieren erfolgen auf der Steuerung. Anhand dieser Verteilung und der Anforderungsmerkmale können die Lösungsmerkmale berechnet werden (siehe Tabelle 9.2).

Tabelle 9.2: Lösungsmerkmale auf Ebene 2 (Ressourcennutzung und Zeitverhalten)

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
		Speicherkompatibilität (load)	
✔	Knoten 1	$2\,\text{MByte}\ (= 2\,097\,152\ \text{Byte}) \leftrightarrow 1024\ \text{Byte} + 1024\ \text{Byte} + 1024\ \text{Byte}$[1]	`IF` $2\,097\,152\ \text{Byte} < 3072\ \text{Byte}$ `THEN` false `ELSE` true
		Speicherkompatibilität (runtime)	
✔	Knoten 1	$15\ \text{KByte}\ (= 15\,360\ \text{Byte}) \leftrightarrow 1024\ \text{Byte} + 1024\ \text{Byte} + 1024\ \text{Byte}$[2]	`IF` $15\,360\ \text{Byte} < 3072\ \text{Byte}$ `THEN` false `ELSE` true
		Klemme-Klemme-Reaktionszeit	
✔	Knoten 1	$\Sigma\ (3\ \text{ms} + 10\ \text{ms} + [2 \times 30\ \text{ms}] + 3\ \text{ms})$	`IF` $76\,\text{ms} < 100\,\text{ms}$ `THEN` true `ELSE` false

[1] Der Wert ist abhängig vom Funktionsblock.
[2] Der Wert ist abhängig vom Funktionsblock.

Der im Rahmen dieses Abschnitts erstellte Steuerungsentwurf einer fertigungstechnischen Anlage zeigt, dass mithilfe der Funktionsmuster geeignete Automatisierungsfunktionen gefunden wurden (siehe Anhang E, Abbildung E.2 auf Seite 169). Des Weiteren wurden diese Automatisierungsfunktionen mithilfe der Verteilungsmuster so verteilt, dass die nfAs (*Ressourcennutzung* und *Zeitverhalten*) erfüllt werden (siehe Anhang E, Abbildung E.4 auf Seite 172 und Tabelle 9.2).

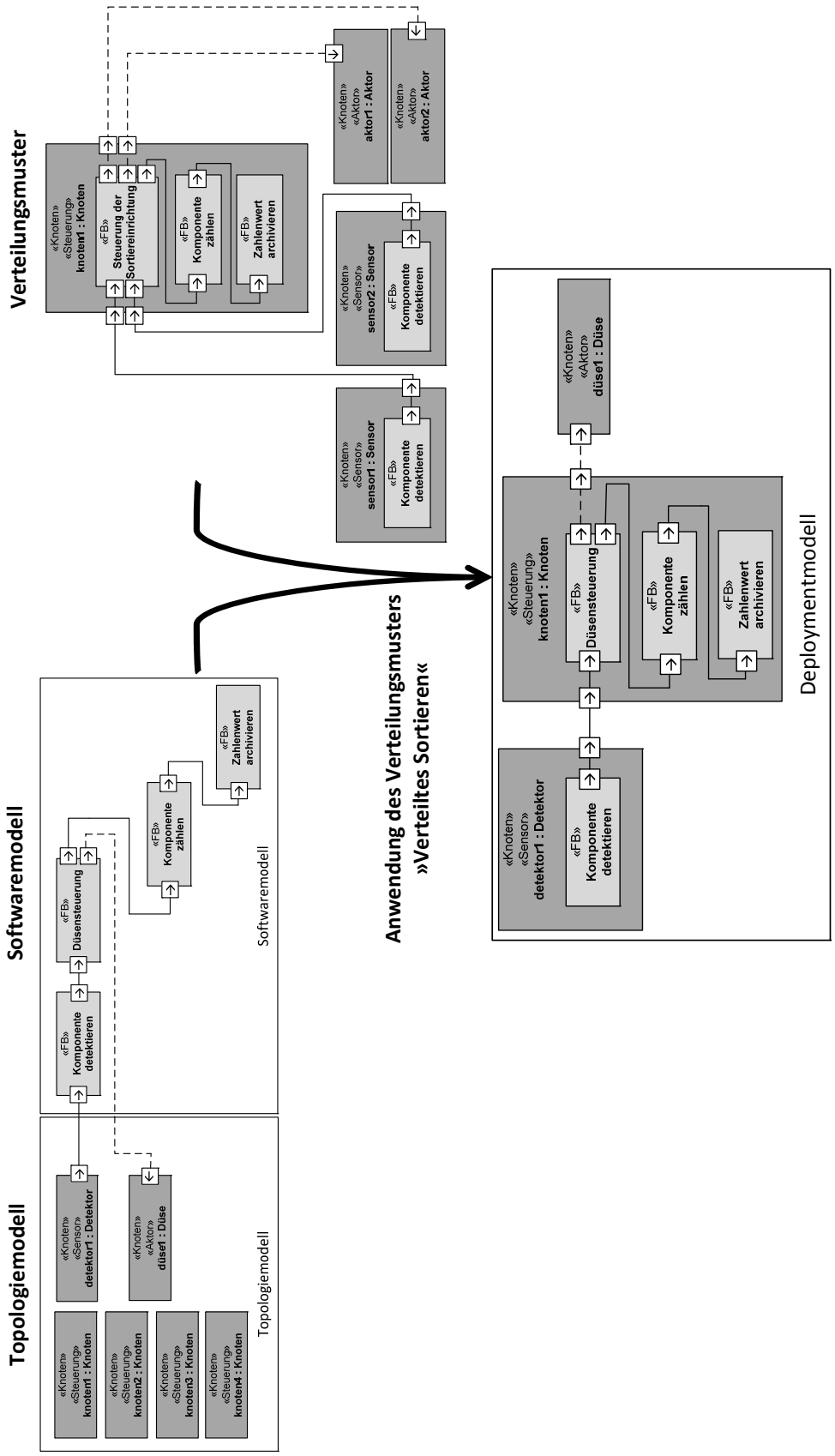

Abbildung 9.4: Anwendung eines Verteilungsmusters

9.2 Verfahrenstechnische Anlage

Im Nachfolgenden wird die Anwendbarkeit des zuvor beschriebenen Konzepts anhand einer verfahrenstechnischen Anlage aufgezeigt. Hierzu erfolgt zum einen eine allgemeine Beschreibung der Anlage, welche die Anforderungen an die Anlage beinhaltet, und zum anderen die Anwendung des Konzepts.

9.2.1 Beschreibung der Anlage

Die im Kontext dieses Kapitels untersuchte *Mischanlage* aus dem Bereich der Verfahrenstechnik dient dem Mischen von zwei Flüssigkeiten und stellt somit ein in der praktischen Anwendung typisches Beispiel dar (siehe Abbildung 9.5 auf der nächsten Seite). Die für das Anwendungsbeispiel betrachtete verfahrenstechnische Anlage soll aus zwei separaten Tanks zwei unterschiedliche Flüssigkeiten in einem Mischtank zusammenführen und dort mischen. Die *Mischanlage* besteht aus fünf einzelnen Behältern sowie einem Mischbehälter, die mittels Rohrleitungen untereinander verbunden sind. Aus den Vorratsbehältern (links: B101, rechts: B201) führen durch je zwei Ventile (links: Y102 und Y301, rechts: Y202 und Y303) Rohrleitungen zu einem zentral angebrachten Mischbehälter (B301). Von dort wird das Gemisch mittels eines Ventils (Y302) und einer Pumpe (N401) in einen Auffangbehälter (B401) weitergeleitet. Das Gemisch wird im nachfolgenden Schritt mittels Ventilen (links: Y801, rechts: Y901, oben: Y402) in die Nachfüllbehälter (links: B801, rechts: B901) weitergeleitet. Damit die Vorratsbehälter (links: B101, rechts: B201) nachgefüllt werden können, kann mithilfe von Pumpen (links: N801, rechts: N901) Flüssigkeit von den Nachfüllbehältern (links: B801, rechts: B901) in die Vorratsbehälter (links: B101, rechts: B201) gepumpt werden. Diese Rohrleitungen sind auch mit steuerbaren Ventilen versehen (links: Y101, rechts: Y201). Alle Tanks sind mit je einem druckmessenden Füllstandsensor versehen (L003, L004, L006, L007, L013 und L015). Zum Durchmischen der zwei Flüssigkeiten verfügt der Mischbehälter (B301) über eine Rührvorrichtung (E003). Abbildung 9.5 zeigt den zuvor beschriebenen schematischen Aufbau der *Mischanlage*, die grundsätzlich aus sechs Behältern, sechs Füllstandsensoren, drei Pumpen, einem Rührer und zehn Ventilen besteht.

Für die Erstellung des Steuerungsentwurfs kann der Anwendungsentwickler zusätzlich zur textuellen, zumeist informellen Spezifikation des Prozesses, auf eine Übersicht der in der Anlage verbauten Sensoren, Aktoren und Steuerungen zurückgreifen (siehe die beiden Tabellen Geräte- und SPS-Datenblatt in Anhang B auf Seite 129). Diese Prozess- und Anforderungsbeschreibung sowie das Geräte- und SPS-Datenblatt sind für die Erstellung der Modelle der *Mischanlage* notwendig, die im nachfolgenden Abschnitt erläutert werden.

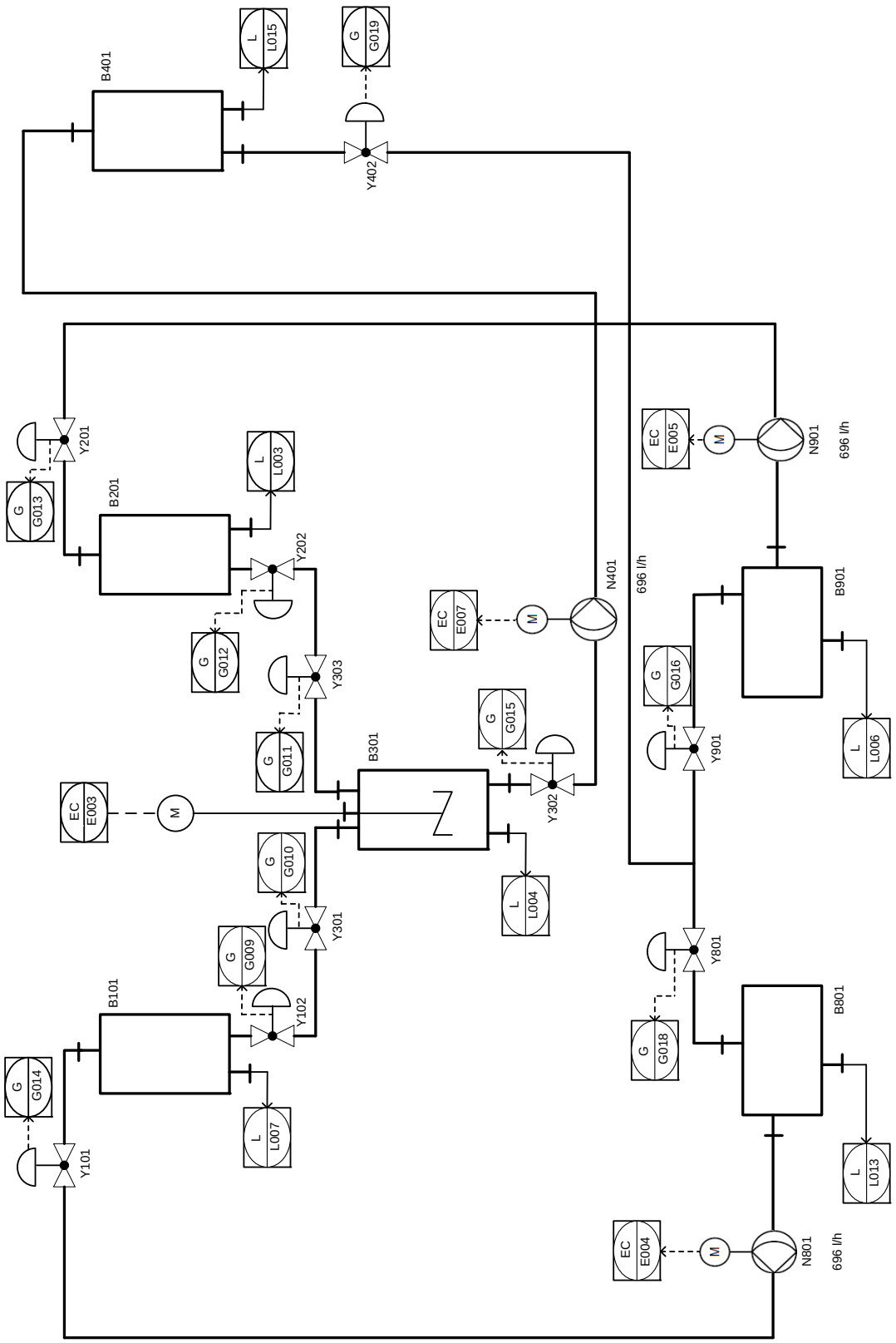

Abbildung 9.5: R&I-Fließbild der Mischanlage (IfA-Forschungsanlage) [vgl. Barth 2011, S. 119]

9.2.2 Spezifikation des Entwurfs

Anhand der in Abschnitt 9.2.1 erläuterten Prozess- und Anforderungsbeschreibung erfolgt im Nachfolgenden die Erstellung eines konzeptionellen Modells für die *Mischanlage*. Dieses Modell beinhaltet – unter Berücksichtigung der gegebenen Anforderungen – alle beschriebenen Funktionalitäten und deren Zusammenhänge mit der notwendigen zu modellierenden Steuerungshardware, den Sensoren und den Aktoren. Durch die Prozess- und Anforderungsbeschreibung der *Mischanlage* kann aufgrund des im Rahmen dieser Arbeit vorgestellten Konzepts ein systematischer und anforderungsgetriebener Entwurf unterstützt werden. Das R&I-Fließbild, beschrieben in Abschnitt 9.2.1, ermöglicht ein allgemeines Verständnis für die an der Planung einer Anlage beteiligten Gewerke, über die Anlage sowie der darin benötigten Funktionalität. Im Anwendungsfall der *Mischanlage* (verfahrenstechnisches Beispiel) wird im Folgenden aufgezeigt, wie der Entwurf verteilter Automatisierungssysteme, basierend auf der in Kapitel 6 beschriebenen Vorgehensweise und der in Kapitel 7 vorgestellten Entwurfsunterstützung, erfolgen kann.

Ebene 4

Auf Basis der in Abschnitt 6.1.2 beschriebenen Aktivitäten auf *Ebene 4* werden auf Grundlage der in Abschnitt 9.2.1 beschriebenen Anforderungen und des beschriebenen Anlagenlayouts die Ergebnisse der *Ebene 4* erstellt. Im Zuge der Automatisierung der *Mischanlage* müssen verschiedene Aspekte berücksichtigt werden. Zum einen wird auf dieser Ebene das für die *Mischanlage* notwendige *Komponentenmodell* erstellt, das die notwendigen Automatisierungsgeräte (zum Beispiel Sensoren und Aktoren) beinhaltet, und zum anderen das *Funktionsmodell*, das die funktionalen Anforderungen untergliedert. Basierend auf dem R&I-Fließbild, können die funktionalen Anforderungen (Anlagenfunktionen) abgeleitet und ermittelt werden. In einem der nachfolgenden Schritte wird das *Komponentenmodell* erstellt (siehe Anhang F, Abschnitt F.1 sowie Abbildung F.1 auf Seite 173), das alle für die *Mischanlage* relevanten Automatisierungsgeräte, wie beispielsweise Pumpe, Ventil, Rührer und Füllstandsensor beinhaltet. Aufgrund der zuvor ermittelten funktionalen Anforderungen können aus einer Vielzahl von Entwurfsmustern die für die aktuelle Problemstellung relevanten Entwurfsmuster (siehe Anhang D) ausgewählt und angewendet werden. Dadurch kann der Anwendungsentwickler bei der Spezifizierung von funktionalen Anforderungen durch automatisierungstechnische Grundfunktionen (Automatisierungsfunktionen) unterstützt werden.

Durch die Anwendung dieser *Funktionsmuster* wird das *Funktionsmodell* der *Mischanlage* erstellt (siehe Anhang F, Abschnitt F.2 sowie Abbildung F.2 auf Seite 174). Diese sogenannten Automatisierungsfunktionen werden in späteren Entwurfsphasen durch Funktionsblöcke realisiert und auf die Hardware verteilt. Die Funktionsmuster beinhalten solche Vorschläge für Automatisierungsfunktionen und sind somit Grundlage für die Erstellung des *Funktionsmodells*. Grundsätzlich besteht die *Mischanlage* aus den Anlagenfunktionen *Transportieren*, *Füllstand überwachen* und *Mischen*.

Exemplarisch werden im weiteren Verlauf die Aktivitäten auf den Ebenen anhand der Anlagenfunktion *Transportieren* der *Mischanlage* dargestellt. Die vollständigen Ergebnisse in puncto *Mischanlage* können den Modellen in Anhang F entnommen werden. Abbildung 9.6 auf der nächsten Seite zeigt somit die Anwendung des *Funktionsmusters* am Beispiel der Anlagenfunktion *Transportieren*.

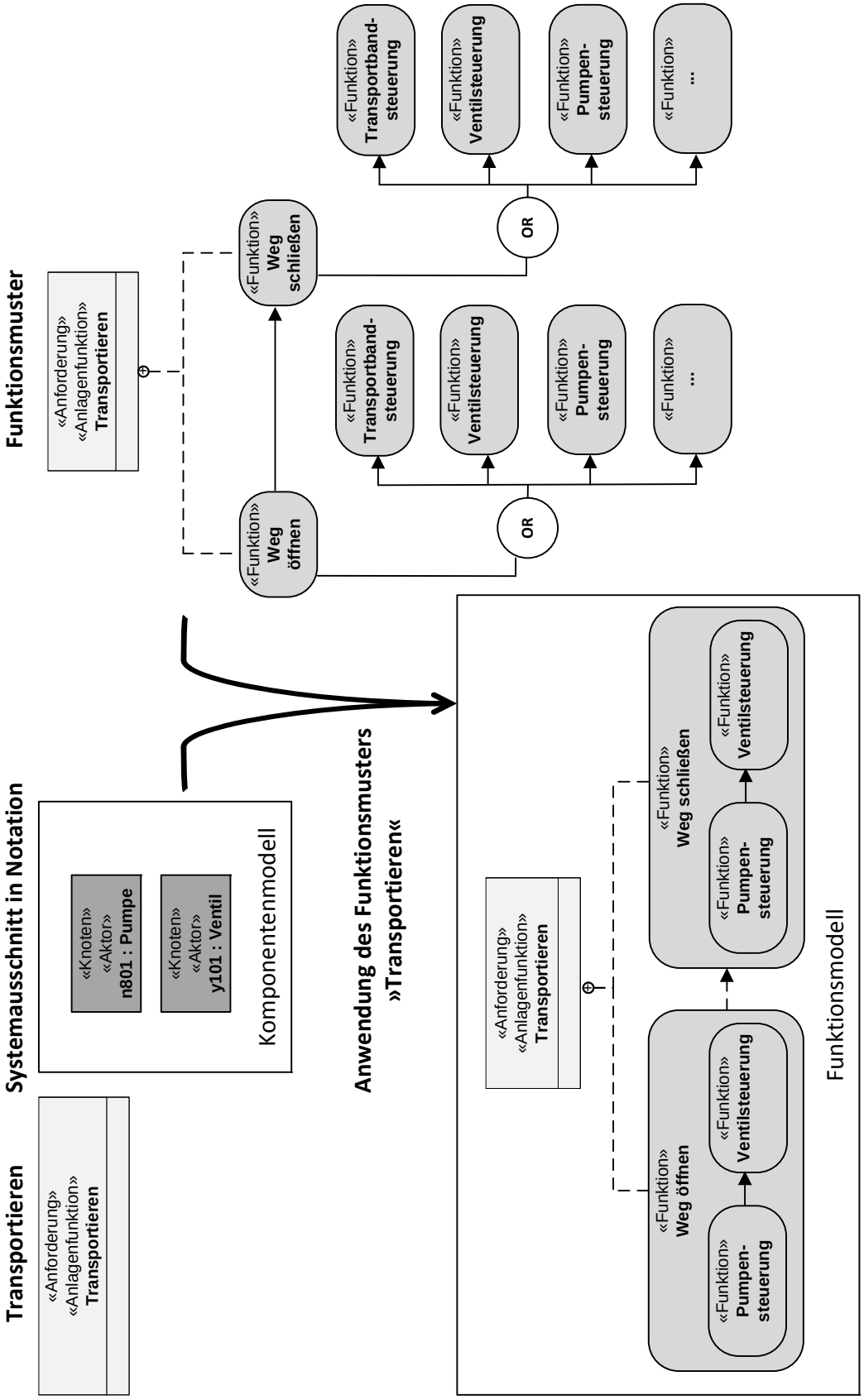

Abbildung 9.6: Anwendung eines Funktionsmusters

Das Funktionsmodell besteht grundsätzlich aus den gleichen Automatisierungsfunktionen wie das Entwurfsmuster. Die speziellen »XOR«- und »OR«-Funktionen des Funktionsmusters werden bei dessen Anwendung näher spezifiziert. Am Beispiel des Funktionsmusters *Transportieren* bedeutet dies, dass die im *Funktionsmuster* mögliche Auswahl von »XOR«-Funktionen bei der Anwendung des Entwurfsmusters durch die Automatisierungsfunktionen *Pumpensteuerung* und *Ventilsteuerung* näher spezifiziert wurde. Das Funktionsmuster *Transportieren* kann abhängig von der Anzahl der Transportwege mehrmals Anwendung finden. Mithilfe des *Komponentenmodells* sowie des R&I-Fließbilds ist ersichtlich, ob für den jeweiligen Transportweg eine Pumpen- und Ventilsteuerung benötigt wird oder lediglich eine Ventilsteuerung.

Ebene 3

Auf Basis der Ergebnisse von *Ebene 4* werden die Aktivitäten auf *Ebene 3* durchgeführt und diese Modelle näher spezifiziert (siehe Abbildung 9.7 auf der nächsten Seite). Im Zuge der Automatisierung der *Mischanlage* müssen verschiedene Aspekte berücksichtigt werden. Zum einen wird auf dieser Ebene das für die *Mischanlage* notwendige *Topologiemodell* erstellt, das die konkreten Automatisierungsgeräte beschreibt, und zum anderen das *Softwaremodell*, das die Spezifizierung der Automatisierungsfunktionen durch ein oder mehrere Funktionsblöcke beinhaltet. Außerdem wird im *Topologiemodell* die Anzahl an Knoten festgelegt und diese werden näher spezifiziert. Des Weiteren werden auf dieser Ebene die benötigten Funktionsblöcke mit der benötigten Automatisierungshardware verknüpft. Die Ein- und Ausgänge der Automatisierungsgeräte der *Mischanlage* können den Datenblättern in Anhang B entnommen werden. Die erstellten Modelle *Topologie- und Softwaremodell* der *Mischanlage* sind in Anhang F, Abschnitt F.3 sowie Abbildung F.3 auf Seite 176 dargestellt.

Abbildung 9.7 zeigt exemplarisch die Aktivitäten und Ergebnisse auf *Ebene 3* am Beispiel der Anlagenfunktion *Transportieren.* In den Aktivitäten auf *Ebene 3* werden die Pumpe, das Ventil und die Knoten unter anderem durch die in den Datenblättern in Anhang B enthaltenen Daten bezüglich Ressourcennutzung, Ein- beziehungsweise Ausgänge sowie des Zeitverhaltens näher beschrieben. Die jeweiligen Ein- und Ausgänge werden im *Topologie- und Softwaremodell* als Eingangs- oder Ausgangsport dargestellt. Diese Merkmale bezüglich Ressourcennutzung und Zeitverhalten sind somit Lösungsmerkmale auf *Ebene 3* (siehe Tabelle 9.3 auf Seite 119), da sie auf dieser Ebene erarbeitet wurden. Tabelle 9.3 enthält die Merkmale der Geräte, der Knoten und der Funktionen. Anhand dieser Merkmale, die den nachfolgenden Ebenen als Anforderungsmerkmale dienen, kann auf *Ebene 2* die Verteilung der Funktionalität erfolgen.

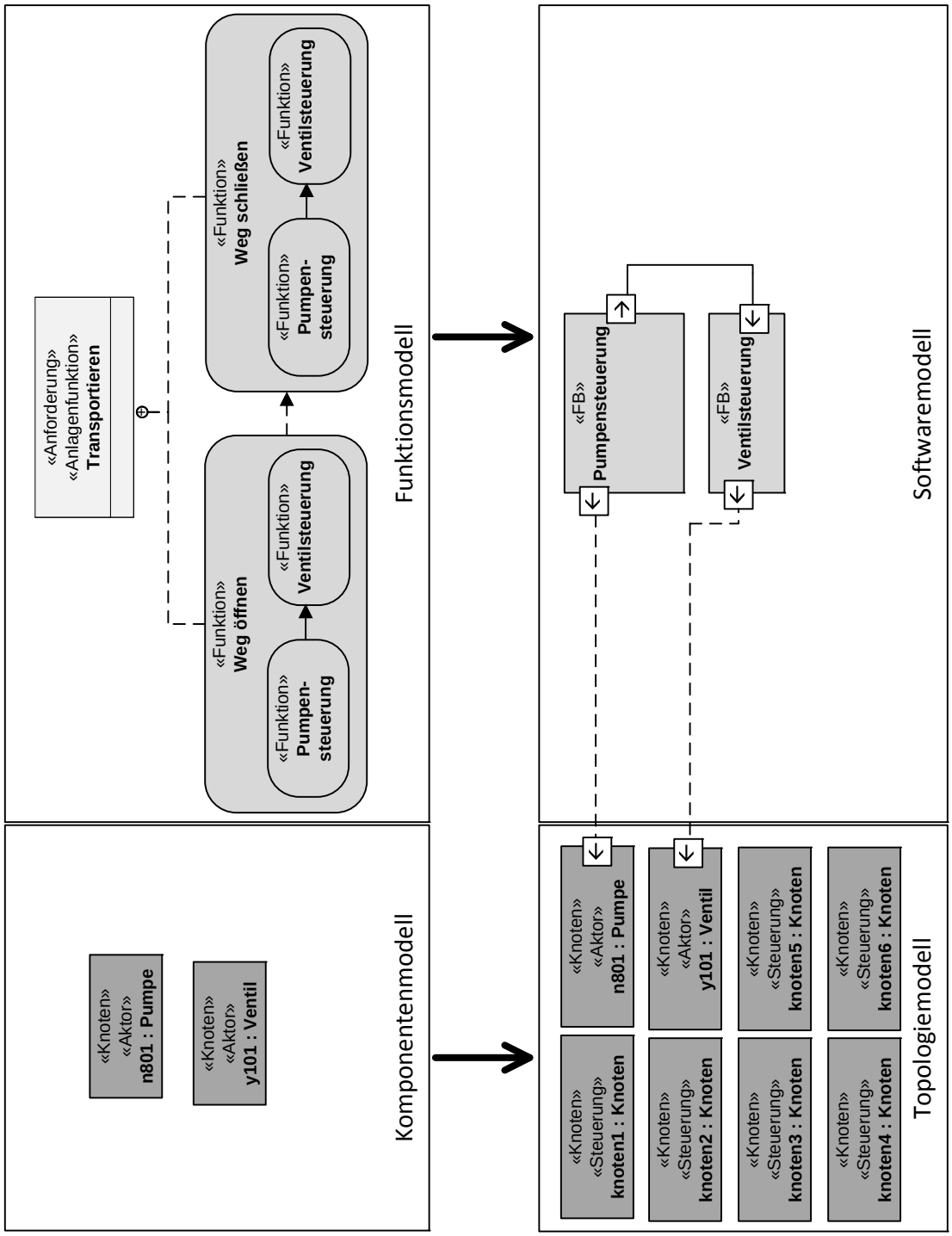

Abbildung 9.7: Aktivität auf Ebene 3 am Beispiel der Anlagenfunktion *Transportieren*

Tabelle 9.3: Lösungsmerkmale auf Ebene 3 (Ressourcennutzung und Zeitverhalten)

Lösungsmerkmale auf Ebene 3	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten 1	50 ms
Knoten 2	50 ms
Knoten 3	50 ms
Knoten 4	50 ms
Knoten 5	50 ms
Knoten 6	50 ms
Gerät-Durchlaufzeit	
Pumpe	3 ms
Ventil	3 ms
Ausführungszeit einer POU	
Pumpensteuerung	10 ms
Ventilsteuerung	10 ms
Feldbuszykluszeit	
Zykluszeit	10 ms
Bereitgestellter Speicher (Runtime)	
Ventil	–
Pumpe	–
Knoten 1	128 KByte
Knoten 2	128 KByte
Knoten 3	128 KByte
Knoten 4	128 KByte
Knoten 5	128 KByte
Knoten 6	128 KByte
Bereitgestellter Speicher (Load)	
Ventil	–
Pumpe	–
Knoten 1	64 KByte
Knoten 2	64 KByte
Knoten 3	64 KByte
Knoten 4	64 KByte
Knoten 5	64 KByte
Knoten 6	64 KByte
Benötigter Speicher (Runtime)	
Pumpensteuerung	1024 Byte
Ventilsteuerung	1024 Byte
Benötigter Speicher (Load)	
Pumpensteuerung	1024 Byte
Ventilsteuerung	1024 Byte

Ebene 2

Auf Basis der Ergebnisse von *Ebene 3* wird die Aktivität auf *Ebene 2* durchgeführt und das *Deploymentmodell* somit näher spezifiziert. Im Zuge der Automatisierung der *Mischanlage* müssen verschiedene Aspekte berücksichtigt werden. Zum einen erfolgt die Verteilung der Funktionalität auf die Hardware und zum anderen muss diese Verteilung so stattfinden, dass die nfAs *Zeitverhalten* und *Ressourcennutzung* erfüllt werden. Die Verteilung der Funktionalität

auf die Hardware wird im *Deploymentmodell* festgehalten. Basierend auf den zuvor ermittelten Automatisierungsfunktionen, können aus einer Vielzahl von Entwurfsmustern die Muster (siehe Anhang D) ausgewählt und angewendet werden, die eine Verteilung der Funktionalität so ermöglichen, dass die nfAs erfüllt werden. Dadurch kann der Anwendungsentwickler bei der Verteilung und der Erfüllung von nfAs unterstützt werden. Durch die Anwendung dieser *Verteilungsmuster* wird das *Deploymentmodell* der *Mischanlage* erstellt (siehe Anhang F, Abschnitt F.4 sowie Abbildung F.4 auf Seite 177).

Abbildung 9.8 auf der nächsten Seite zeigt somit die Anwendung des *Verteilungsmusters* am Beispiel *Transportieren*. In der Aktivität auf *Ebene 2* werden die Lösungsmerkmale von *Ebene 3* zu Anforderungsmerkmalen, mithilfe derer die Lösungsmerkmale von *Ebene 2* berechnet werden können. Hierzu ist die Verteilung der Funktionalität zu berücksichtigen. Die Funktionalität der Pumpensteuerung und der Ventilsteuerung wird auf einen Knoten verteilt, der mit der jeweiligen Pumpe beziehungsweise dem jeweiligen Ventil kommuniziert. Anhand dieser Verteilung und den Anforderungsmerkmalen können die Lösungsmerkmale berechnet werden (siehe Tabelle 9.4).

Tabelle 9.4: Lösungsmerkmale auf Ebene 2 (Ressourcennutzung und Zeitverhalten)

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
		Speicherkompatibilität (load)	
✔	Knoten 1	64 KByte $\left(= 65\,536 \text{ Byte}\right) \leftrightarrow 1024 \text{ Byte} + 1024 \text{ Byte}$[1]	`IF` 65 536 Byte $<$ 2048 Byte `THEN` false `ELSE` true
		Speicherkompatibilität (runtime)	
✔	Knoten 1	128 KByte $\left(= 131\,072 \text{ Byte}\right) \leftrightarrow 1024 \text{ Byte} + 1024 \text{ Byte}$[2]	`IF` 131 072 Byte $<$ 2048 Byte `THEN` false `ELSE` true
		Klemme-Klemme-Reaktionszeit	
✔	Knoten 1	$\Sigma\left(3 \text{ ms} + 50 \text{ ms} + \left[2 \times 10 \text{ ms}\right] + 3 \text{ ms}\right)$	`IF` 76 ms $<$ 100 ms `THEN` true `ELSE` false

[1] Der Wert ist abhängig vom Funktionsblock.
[2] Der Wert ist abhängig vom Funktionsblock.

Der im Rahmen dieses Abschnitts erstellte Steuerungsentwurf einer verfahrenstechnischen Anlage zeigt, dass mithilfe der Funktionsmuster geeignete Automatisierungsfunktionen gefunden wurden (siehe Anhang F, Abbildung F.2 auf Seite 174). Des Weiteren wurden diese Automatisierungsfunktionen mithilfe der Verteilungsmuster so verteilt, dass die nfAs (*Ressourcennutzung* und *Zeitverhalten*) erfüllt werden (siehe Anhang F, Abbildung F.4 auf Seite 177 und Tabelle 9.4).

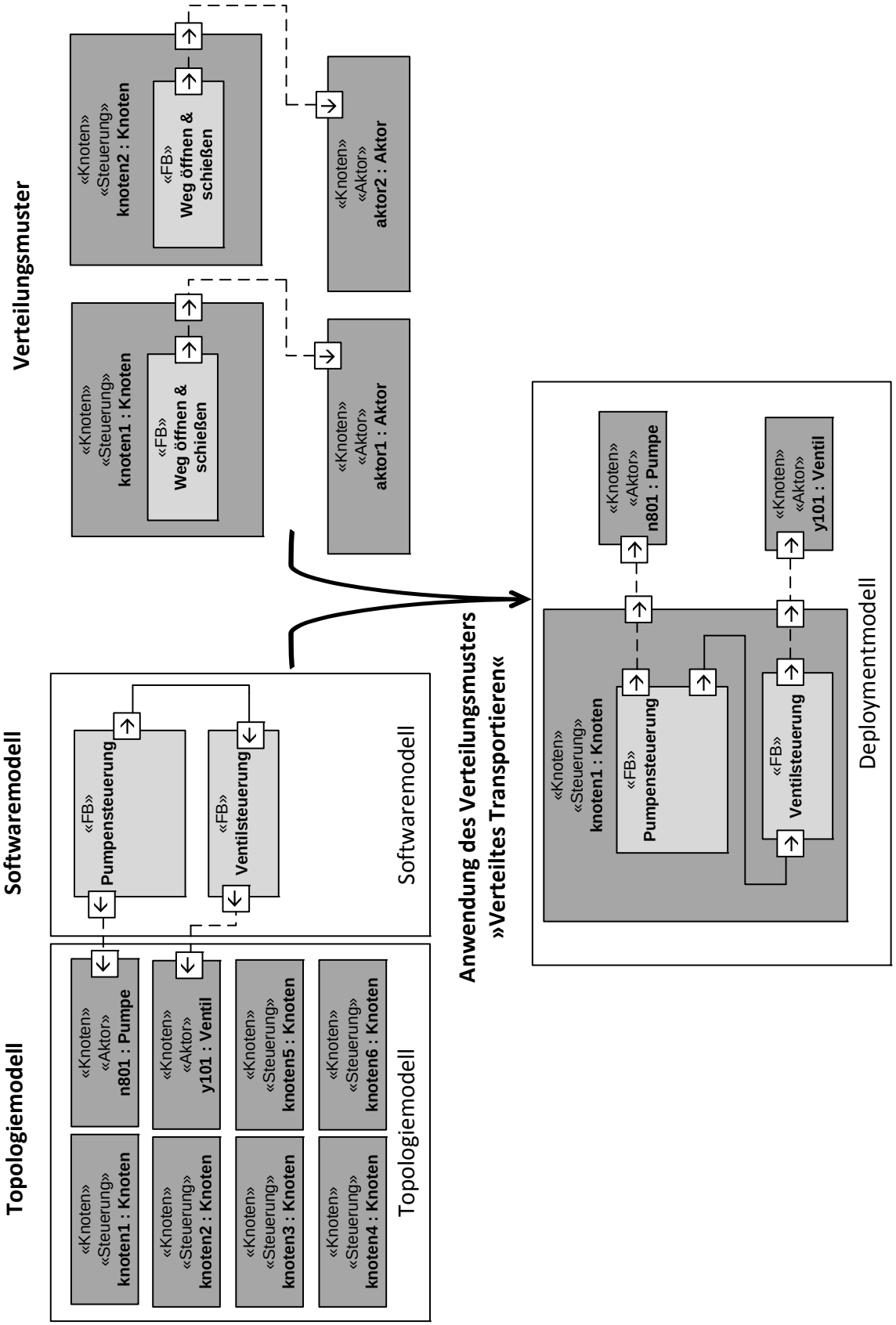

Abbildung 9.8: Anwendung eines Verteilungsmusters

10 Zusammenfassung und Ausblick

Als Abschluss der Arbeit werden in diesem Kapitel die wesentlichen Ergebnisse und Erkenntnisse resümiert. Ein Ausblick auf weitere zukünftige Entwicklungen und Forschungsfragen im Kontext des vorgestellten Konzepts runden das Kapitel ab.

10.1 Zusammenfassung

Das Engineering heutiger Automatisierungssysteme ist zunehmend geprägt durch immer komplexere Prozesse und Abläufe, resultierend aus der fortschreitenden Entwicklung technischer Möglichkeiten sowie einer voranschreitenden Individualisierung von Produkten. Die Automatisierungssysteme, die diese Prozesse steuern und kontrollieren, müssen an diese Anforderungen angepasst werden und entwickeln sich in ähnlich rasanter Weise. Mit dem Einsatz von verteilten Automatisierungssystemen soll den Herausforderungen an moderne Automatisierungssysteme begegnet werden. Moderne und komplexe Automatisierungssysteme implizieren, dass eine Vielzahl von Daten und Signalen verarbeitet und ausgewertet werden muss, was jedoch in Kombination mit einer heutzutage üblichen intuitiven Vorgehensweise und einer direkten Implementierung zu fehler- und störanfälligen Automatisierungssystemen führt. Die Ursachen, die Automatisierungssysteme fehler- und störanfälliger machen, werden zumeist erst während des Anlagenbetriebs entdeckt und können auf die Entwurfsphase eines Automatisierungssystems zurückgeführt werden. Ihre Beseitigung verursacht oftmals essenzielle Mehrkosten.

Ist die Planungsphase des Engineerings abgeschlossen, beginnt der Steuerungsentwurf, in dem die Hardware und die Software zusammengefügt werden müssen. Insbesondere bei verteilten Automatisierungssystemen stellt sich die Frage, wie die Steuerungslogik auf die Steuerungshardware verteilt werden soll. Eine systematische Vorgehensweise in Zusammenhang mit einer Wiederverwendung von bewährten Lösungen in Form von Entwurfsmustern bildet eine adäquate Basis für die Unterstützung des systematischen Entwurfs von verteilten Automatisierungssystemen. Dennoch haben in der Automatisierungstechnik die Entwurfsmuster in der praktischen Anwendung bisher wenig Anerkennung gefunden. Die ersten Ansätze von Entwurfsmustern in der Automatisierungstechnik beziehen sich zumeist auf einige wenige Problemstellungen und berücksichtigen keine verteilten Automatisierungssysteme. Für die Verteilung von Funktionalität werden oftmals Algorithmen eingesetzt, die aufgrund ihrer Komplexität eine Einbindung von nfAs nur bedingt erlauben.

Der Einsatz von *Entwurfsmustern* während des Anlagenentwurfs automatisierungstechnischer Systeme unterstützt den Anwendungsentwickler bei der hierarchischen Zerlegung von funktionalen Anforderungen – Anlagenfunktionen – in automatisierungstechnische Grundfunktionen – Automatisierungsfunktionen – und bei der Verteilung von Steuerungssoftware auf die Steuerungshardware insoweit, dass bei der Verteilung nicht-funktionale Anforderungen (nfAs) erfüllt werden. Hierzu wurde im Rahmen der vorliegenden Arbeit eine Methodik für den systematischen Entwurf verteilter Automatisierungssysteme vorgestellt, mit dem auf Basis eines Vorgehensmodells und unter Anwendung von Entwurfsmustern eine Funktionsverteilung vorgenommen werden kann. Hierbei wurden einerseits die besonderen funktionalen und nicht-funktionalen Anforderungen, die sich aus der Verteilung der Steuerungssoftware ergeben, berücksichtigt und andererseits wurde die Wiederverwendung von guten Lösungen gefördert.

Ziel dieser Arbeit war die Auswahl eines prinzipiellen Schemas für die Verteilung und Koordination der Softwarefunktionalität und nicht deren konkrete Implementierung. Des Weiteren fokussiert die Arbeit auf Auswahl und Verteilung der Funktionalität; die Auswahl der benötigten Hardware wurde nicht betrachtet. Die im Rahmen der vorliegenden Arbeit erzielten Ergebnisse bezüglich der angewandten Methoden – Vorgehensweisen – sowie der Wiederverwendung werden im Nachfolgenden zusammengefasst.

Methode

Der systematische Entwurf verteilter Automatisierungssysteme wurde im Rahmen dieser Arbeit durch eine methodische Vorgehensweise erweitert. Hierzu wurden zwei Kernelemente beschrieben, ein Vorgehensmodell sowie die Berücksichtigung von nfAs in diesem. Das im Rahmen dieser Arbeit vorgestellte Vorgehensmodell sowie die Einordnung der nfAs wurden in Zusammenarbeit mit dem *Lehrstuhl für Automatisierung und Informationssysteme* (AIS) der Technischen Universität München und dem *Institut für Automatisierungstechnik* (IFAT) der Otto-von-Guericke-Universität Magdeburg entwickelt. Das Vier-Ebenen-Vorgehensmodell, basierend auf dem V-Modell, ermöglicht einen systematischen Entwurf und ist speziell auf verteilte Automatisierungssysteme ausgerichtet. Es erfolgte die Einordnung von nfAs in das Vorgehensmodell, sodass ein systematischer und anforderungsgetriebener Entwurf verteilter Automatisierungssysteme unterstützt wird. Dieses Vorgehen hat zum Ziel, dass die nfAs auf der jeweils relevanten Ebene des Vorgehensmodells bearbeitet werden können. Diese ebenenweise Vorgehensweise und die Einordnung von nfAs in das Vorgehensmodell erlauben eine Berücksichtigung von nfAs, basierend auf quantitativ erfassbaren Merkmalen.

Wiederverwendung

Auf Basis des Vorgehensmodells wurde dem Ziel der Wiederverwendung dadurch Rechnung getragen, dass im Rahmen dieser Arbeit ein Musterkonzept entwickelt wurde, das den Anwendungsentwickler beim Entwurf mittels Entwurfsvorschlägen unterstützt. Das erstellte Musterkonzept beschreibt zwei Kernelemente der Entwurfsunterstützung, die »Funktionsmuster« und die »Verteilungsmuster«. Die Funktionsmuster unterstützen den Anwendungsentwickler beim Entwurf der Funktionalität dadurch, dass für die gewünschte und spezifizierte Funktionalität der Anlage jeweils bewährte und typische Automatisierungslösungen angeboten werden. Des Weiteren erfährt der Anwendungsentwickler dahingehend Unterstützung, dass *Verteilungsmuster* verschiedene Verteilungsalternativen für die Verteilung der Funktionalität auf Automatisierungsgeräte, wie beispielsweise Sensoren, Aktoren und Steuerungen, bereitstellen und die gewünschten nfAs erfüllen.

Evaluation

Durch zwei Anwendungsbeispiele aus dem Bereich der Fertigungs- und Verfahrenstechnik wurde die Anwendbarkeit und Schlüssigkeit des Lösungskonzepts nachgewiesen. Hierbei wurde die Eignung dieses Ansatzes, den Entwurfsprozess von verteilten Automatisierungssystemen zielgerichtet zu leiten und zu unterstützen, nachgeprüft. Des Weiteren wurde das entwickelte Konzept hinsichtlich praktischer Eignung durch verschiedene Nutzergruppen systematisch evaluiert. Die Nutzergruppe, welche durch *Entwurfsmuster* unterstützt wurde, musste aus einer Vielzahl von *Funktions-* und *Verteilungsmustern* die auswählen, die den funktionalen und bei der Verteilung den nicht-funktionalen Anforderungen entsprechen. Dadurch wurde der statistische Nachweis erbracht, dass das entwickelte Konzept einen Mehrwert erzeugt (siehe die nachfolgende Tabel-

le 10.1). Die Untersuchung des Aspekts »Verteilung«, ein wesentlicher Teilaspekt des Entwurfs verteilter Steuerungsarchitekturen, zeigte in der Evaluation eine qualitativ deutliche Verbesserung durch den Einsatz von Mustern in Kombination mit Merkmalen und Notation im Vergleich zu den Evaluationen mit IEC 61 131-3, IEC 61 499 sowie den Merkmalen und der Notation ohne Muster. Durch die Evaluation konnte gezeigt werden, dass das entwickelte Konzept auf Basis eines an die Domäne angepassten Vorgehensmodells sowie durch Muster eine deutliche Verbesserung im Aspekt »Verteilung von Funktionalität« für den Anwendungsentwickler ermöglicht.

Tabelle 10.1: Evaluationsergebnisse

Unteraufgabe	Gruppe	Teilnehmer	Mittelwert[1]	Standardabweichung
Softwarefunktionalität	IEC 61 131-3	4	11,0	8,72
	IEC 61 499	4	17,75	3,4
	FAVA Merkmale	5	20,0	6,63
	FAVA Muster	4	25,8	2,6
Verteilung	IEC 61 131-3	4	11,25	10,44
	IEC 61 499	4	16,5	4,51
	FAVA Merkmale	5	9,8	3,35
	FAVA Muster	4	21,0	14,4

[1] Höherer Zahlenwert (höhere Punktzahl) – auch bei der Standardabweichung – bedeutet besseres Ergebnis. Statistische Unsicherheiten können auftreten.

Insgesamt kann hieraus der Schluss gezogen werden, dass der Ansatz, der im Rahmen dieser Arbeit entwickelt wurde, geeignet ist, den Anwendungsentwickler mit einer Vorgehensweise zu führen, die es ihm ermöglicht, Auswirkungen von Designentscheidungen im Einzelnen selbst abschätzen zu können, mit der Möglichkeit, bei der jeweiligen Entscheidung Hilfestellung zu erlangen. Dadurch kann die Erfüllung von nfAs schon während des Entwurfs berücksichtigt werden. Das Entwurfsergebnis, das die für die Anwendung benötigen Funktionsblöcke und deren Laufzeitumgebung enthält, kann dann in einer der Sprachen der IEC 61 131-3 programmiert werden.

Möglichkeiten und Grenzen des Entwurfsmusterkonzepts

Die vorliegende Arbeit beschreibt ein Konzept, in dem die Verteilung der Automatisierungsfunktionen und die Berechnung der nfAs vom Anwendungsentwickler durchgeführt werden muss. Eine automatisierte Verteilung und Berechnung der nfAs ist in diesem Konzept nicht vorgesehen. Des Weiteren erläutert die vorliegende Arbeit *Entwurfsmuster* für zwei von vier Ebenen des Vorgehensmodells. Eine Entwurfsunterstützung für die zwei fehlenden Ebenen ist im Konzept nicht vorgesehen, weil dies Implementierungsdetails beinhalten würde. Es wäre jedoch zu untersuchen, ob für diese zwei Ebenen eine Entwurfsunterstützung möglich ist. Um einen produktiven Einsatz des vorgestellten Konzepts zu ermöglichen, sollte ein Musterkatalog für alle Anlagenfunktionen und deren Automatisierungsfunktionen sowie deren Verteilung bereitgestellt werden, was nur von erfahrenen Anwendungsentwicklern geleistet werden kann.

10.2 Ausblick

Die vorliegende Arbeit stellt die Basis für einen systematischen und anforderungsgetriebenen Entwurf von verteilten Automatisierungssystemen sowie deren Entwurfsunterstützung bereit. Das Konzept bietet die Möglichkeit, ein Vorgehensmodell zu nutzen, um verteilte Steuerungsarchitekturen aufzubauen. Durch die Anwendung von Entwurfsmustern wird eine geeignete Entwurfsunterstützung aufgezeigt, die den Anwendungsentwickler leitet.

Die prototypische Umsetzung des Konzepts ist in einer Grundfunktionalität implementiert und ermöglicht die Bereitstellung der Entwurfsmuster in den unterschiedlichen Ebenen in Form einer Bibliothek. Dadurch stehen dem Anwendungsentwickler die jeweiligen Entwurfsmuster zur Anwendung zur Verfügung. Für einen Produktiveinsatz muss diese prototypische Umsetzung erweitert werden, um eine automatisierte Verteilung, basierend auf den Verteilungsmustern, durchführen zu können. Erste konzeptionelle Ansätze für diese automatische Verteilung sind in dieser Arbeit dargestellt.

Die im Rahmen dieser Arbeit entwickelte Methode fokussiert nach derzeitigem Stand auf die Neuentwicklung eines verteilten Automatisierungssystems. Inwieweit die in dieser Arbeit beschriebenen Methoden und Vorgehensweisen für die Integration von Änderungen – auch Strukturänderungen – in bestehenden Automatisierungssystemen Anwendung finden können, wäre Gegenstand einer weiteren Untersuchung. Generell würde damit die Möglichkeit bereitgestellt, bestehende Strukturen von Automatisierungssystemen zu ändern, um dadurch verteilte Automatisierungsstrukturen aufzubauen und durch die Entwurfsunterstützung den Anwendungsentwickler bei der Verteilung der Funktionalität zu unterstützen.

Zukünftig ist es darüber hinaus denkbar, den Ansatz um weitere Phasen des V-Modells – rechter Ast – zu erweitern, der hauptsächlich die Testbarkeit des erstellten Konzepts beinhaltet. Inwieweit sich das in dieser Arbeit beschriebene Konzept der Vorgehensweise und Entwurfsunterstützung für die Erweiterung um bekannte, neue oder frühere Testmethoden eignet, wäre ebenfalls Gegenstand weiterer Forschung.

Der Bereich Informatik hat zur Erfüllung des Aspekts Vernetzung unternehmens- und domänenübergreifender Dienste beziehungsweise Aufgaben Cyber-Physical Systems (CPS) entwickelt. Erste Ansätze zur Übertragung dieses Aspekts auf die Automatisierungstechnik beschreiben das Potenzial und mögliche Anwendungsgebiete von CPS in der Automatisierungstechnik. Das in dieser Arbeit entwickelte Konzept fokussiert auf verteilte Steuerungsarchitekturen von Systemen und Anlagen und interagiert nicht über Unternehmensgrenzen hinweg. Die Erweiterung des Konzepts für diesen Untersuchungsgegenstand wäre Aufgabe weiterer Forschungsarbeiten. Die zuvor beschriebenen möglichen Erweiterungen des im Rahmen dieser Arbeit beschriebenen Konzepts bieten interessante Weiterentwicklungen und Optimierungen.

Anhang

Anhang A Vorgehensmodelle

Die folgende Tabelle zeigt das NA Arbeitsblatt 35, das eine Handlungsempfehlung für die chemische Verfahrenstechnik ist, aber auch für andere PLS verwendet werden kann.

Ziel						
Die ausführbare Anlage	Die genehmigungsfähige Anlage	Die ausschreibbare Anlage	Die errichtbare Anlage	Die funktionsfähige Anlage	Die produktionsfähige Anlage	Die bewertete und abgerechnete Anlage
Projektierung						
1. Grundlagenermittlung	2. Vorplanung	3. Basisplanung	4. Ausführungsplanung	5. Errichtung	6. Inbetriebsetzung	7. Projektabschluss
1.1 Projektziele festlegen 1.2 Grobkosten schätzen	2.1 Anlagenkonzept festlegen 2.2 Kosten schätzen 2.3 Kosten kalkulieren – 3.4	3.1 PLT-Funktionen festlegen 3.2 Verfahrenstechnische Daten beschaffen 3.3 technische Realisierung festlegen 3.4 Kosten kalkulieren	4.1 Geräte festlegen 4.2 Zentrale Einrichtungen festlegen 4.3 Leitsystem spezifizieren 4.4 Stellenpläne erzeugen 4.5 Stellenfunktionspläne erzeugen 4.6 Montageunterlagen erstellen	5.1 Bestellung veranlassen 5.2 Lieferung bestätigen 5.3 Software konfigurieren 5.4 Montage vorbereiten 5.5 Montage überwachen 5.6 Funktion prüfen	6.1 Personal ausbilden 6.2 Inbetriebsetzung untersützen 6.3 Dokumentation revidieren 6.4 Dokumentation überwachen	7.1 Abschlussbericht erstellen 7.2 Projektrechnung erstellen
Qualitätsmanagement						
Projektmanagement						

Tabelle A.1: Standardstrukturplan der NA 35 [vgl. NA 35 2003, S. 6][#]

Die folgende Abbildung zeigt das Vorgehensmodell der VDI 2206-Richtlinie für eine flexible Vorgehensweise zur Entwicklung mechatronischer Systeme.

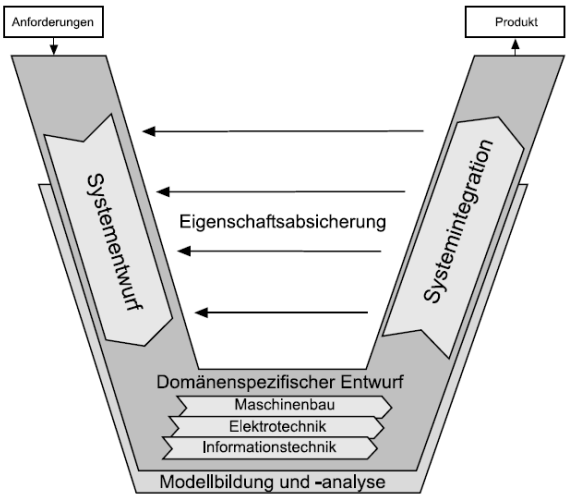

Abbildung A.1: V-Modell als Makrozyklus [VDI 2206 2004, S. 29][#]

Die folgende Abbildung zeigt das V-Modell, ein international anerkanntes Vorgehensmodell für die Projektabwicklung.

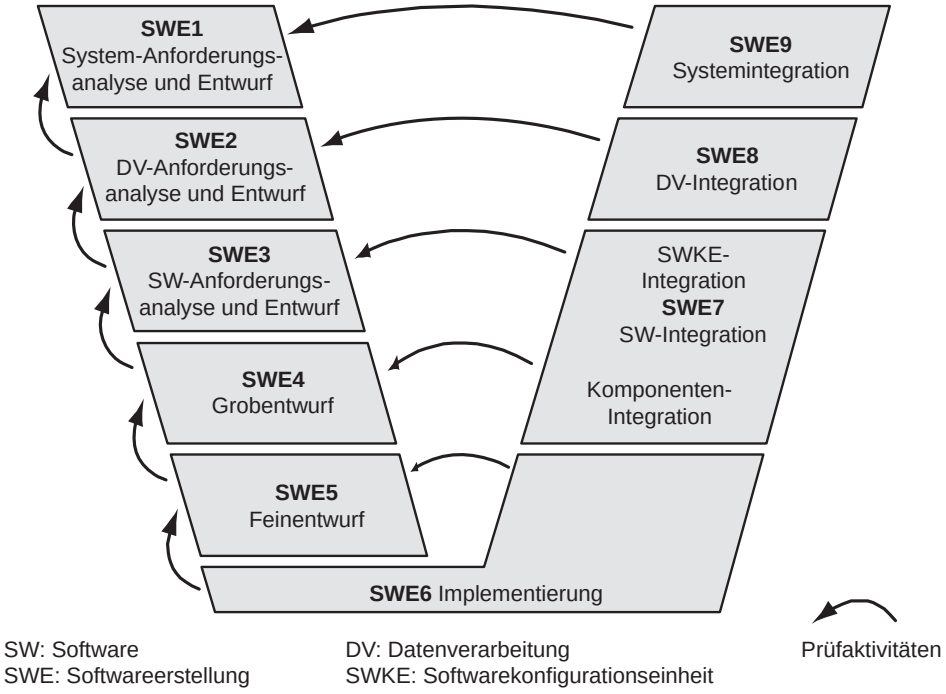

Abbildung A.2: V-Modell [Bender 2005, S. 32]

Die folgende Abbildung zeigt das V-Modell xt, eine Weiterentwicklung des V-Modells und eine Vorgehensweise für die Planung und Durchführung von Projekten.

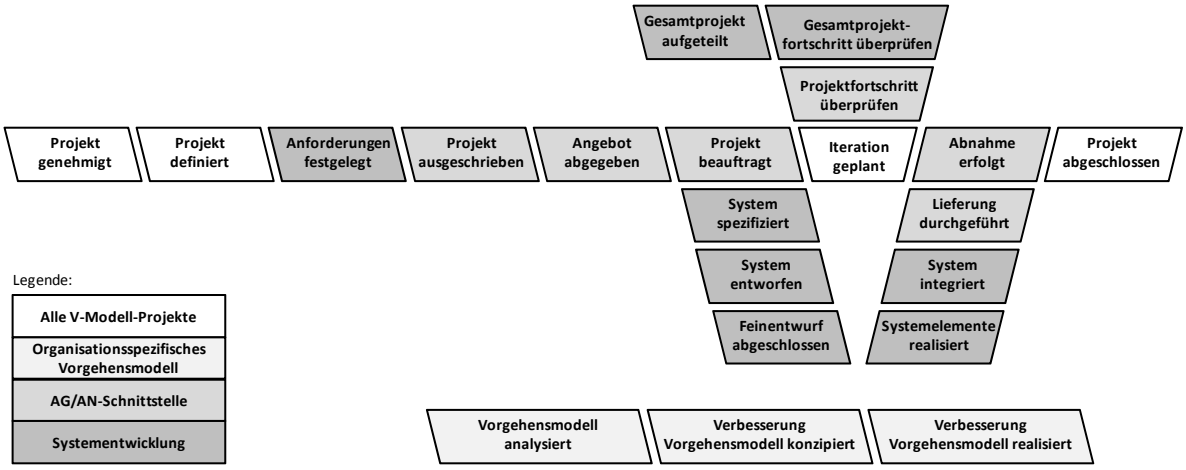

Abbildung A.3: V-Modell xt [vgl. Rausch & Broy 2008, S. 12]

Die folgende Abbildung zeigt das 3-Ebenen-Vorgehensmodell, ein Vorgehensschema zur Entwicklung mechatronischer Produkte.

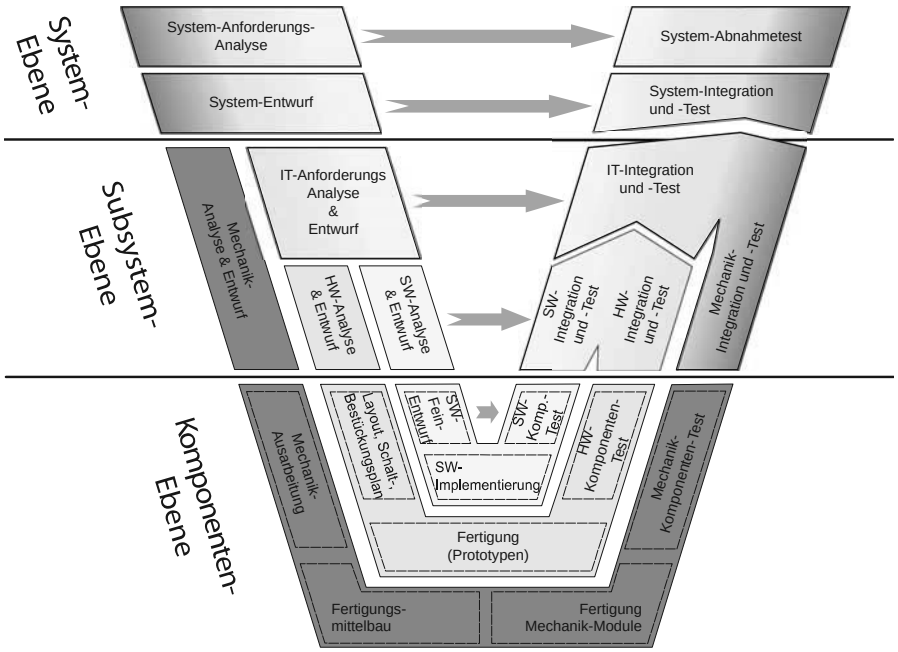

Abbildung A.4: 3-Ebenen-Vorgehensmodell [Bender 2005, S. 45]

Anhang B Datenblätter für die Anwendungsbeispiele

Im Nachfolgenden werden die Datenblätter für das fertigungstechnische Beispiel (Abschnitt 9.1) dargestellt.

Gerätedatenblatt

Tabelle B.1: Gerätedatenblatt für die Metalltrennungsanlage

Motor	Allgemeines	Anschluss Speicher	Profibus 0 KByte
	Input	var_start var_speed	Bool 1/0 (an/aus) Int 0–1000 (0 bis 10 km/h)
Temperatursensor	Allgemeines	Anschluss Speicher	Analog 10 KByte
	Output	var_temperatur	Int 0–150 (−50 °C bis 100 °C)
Detektor	Allgemeines	Anschluss Speicher	Analog 10 KByte
	Output	var_teilerkannt	Bool 1/0 (an/aus)
	Zeitverhalten	Zykluszeit Übergangszeit	2 ms 1 ms
Düse	Allgemeines	Anschluss Speicher	24 V/Profibus 0 KByte
	Input	var_an	Bool 1/0 (an/aus)
	Zeitverhalten	Zykluszeit Übergangszeit	2 ms 1 ms
Füllstandsensor	Allgemeines	Anschluss Speicher	Profibus 40 MByte
	Output	var_level	Float 0.0 – 100.0 (0 % – 100 %)
Hupe	Allgemeines	Anschluss	Analog
	Input	var_an	Bool 1/0 (an/aus)

SPS-Datenblatt

Tabelle B.2: SPS-Datenblatt für die Metalltrennungsanlage

S7-300 CPU 312	Anschluss	Profibus, Ethernet, Analog
	Bereitgestellter Speicher (Runtime memory)	15 KByte
	Bereitgestellter Speicher (Load memory)	2 MByte
	Zykluszeit	50 ms
S7-300 CPU 317	Anschluss	Profibus, Ethernet, Analog
	Bereitgestellter Speicher (Runtime memory)	19 KByte
	Bereitgestellter Speicher (Load memory)	8 MByte
	Zykluszeit	100 ms
Feldbus	Zykluszeit	30 ms

Im Nachfolgenden werden die Datenblätter für das verfahrenstechnische Beispiel (Abschnitt 9.2) dargestellt.

Gerätedatenblatt

Tabelle B.3: Gerätedatenblatt für die Mischanlage

Motor Mischer	Allgemeines	Anschluss Speicher	Profibus 0 KByte
	Input	var_start	Bool 1/0 (an/aus)
	Zeitverhalten	Zykluszeit	3 ms
Motor Pumpe	Allgemeines	Anschluss Speicher	Profibus 0 KByte
	Input	var_start	Bool 1/0 (an/aus)
	Zeitverhalten	Zykluszeit	3 ms
Ventil	Allgemeines	Anschluss Speicher	Profibus 0 KByte
	Input	var_auf	Bool 1/0 (an/aus)
	Zeitverhalten	Zykluszeit	3 ms
Füllstandsensor	Allgemeines	Anschluss Speicher	Profibus 10 KByte
	Output	var_level	Float 0.0 – 100.0 (0 % – 100 %)
	Input	var_start	Bool 1/0 (an/aus)

SPS-Datenblatt

Tabelle B.4: SPS-Datenblatt für die Mischanlage

750–833	Anschluss	Profibus, Ethernet, Analog
	Bereitgestellter Speicher (Runtime memory)	128 KByte
	Bereitgestellter Speicher (Load memory)	64 KByte
	Zykluszeit	50 ms
Feldbus	Zykluszeit	10 ms

Anhang C Entwurfsmuster der fördertechnischen Anlage

Anhang C 1 Funktionsmuster

Im Folgenden sind die Funktionsmuster für die fördertechnische Anlage (siehe Abschnitt 9.1) dargestellt.

Anhang C 1.1 Funktionsmuster »Transportieren«

Im Folgenden ist das *Funktionsmuster* für die Anlagenfunktion *Transportieren* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Transport

Mustername: Transportieren von Komponenten exklusiv Wegverfolgung und Reservierung

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Transportieren* zu erfüllen. In diesem Entwurfsmuster wird lediglich die Ressourcenansteuerung betrachtet. Die Automatisierungsfunktionen für die Wegverfolgung – Weg suchen und Weg wählen – und für die Reservierung – Weg reservieren und Reservierung aufheben – sind in diesem Entwurfsmuster nicht enthalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungstechnik und in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- für den Transport nicht mehrere Transportwege zur Verfügung stehen.
- das Suchen und Reservieren von Ressourcen nicht benötigt wird, da auf die Ressourcen nicht gleichzeitig zugegriffen wird.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

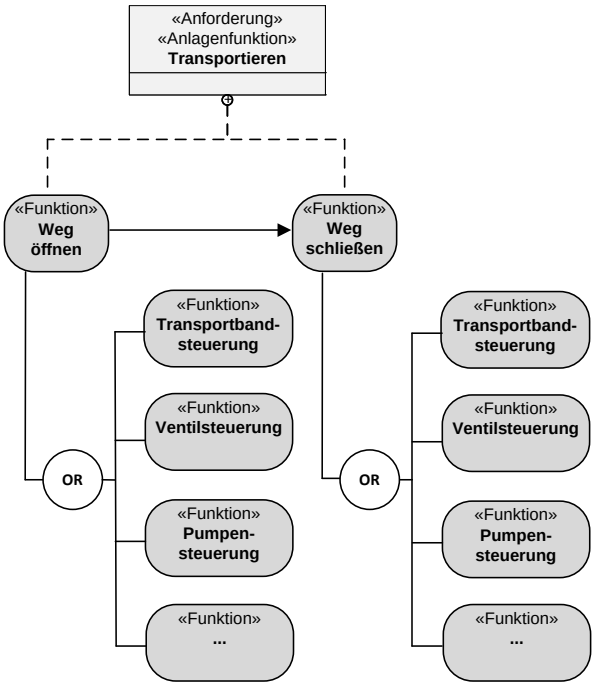

Abbildung C.1: Funktionsmuster »Transportieren ohne Reservierung«

Teilnehmer: Dieses Funktionsmuster besteht aus zwei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden.

1. *Weg öffnen:* Direktes Ansteuern der benötigten Ressourcen ohne zusätzliche Reservierung. Dies bedeutet, dass der Weg bei Bedarf für den Transport geöffnet wird. Zum Öffnen von Ressourcen, wie beispielsweise Anschalten von Förderbändern oder Öffnen von Ventilen, stehen mehrere Verfahren zur Verfügung. Hierbei ist zu beachten, dass die jeweiligen Ressourcen zum einen lediglich angeschaltet beziehungsweise geöffnet und zum anderen mit bestimmten Parametern, wie beispielsweise Geschwindigkeit, angeschaltet beziehungsweise geöffnet werden können.
2. *Weg schließen:* Nach erfolgreichem Transportieren der Komponenten muss der Transportweg geschlossen werden. Dies bedeutet, dass beispielsweise Förderbänder ausgeschaltet und Ventile geschlossen werden. Hierzu stehen die gleichen Verfahren wie beim Weg öffnen zur Verfügung.

Anhang C 1.2 Funktionsmuster »Not-Halt«

Im Nachfolgenden ist das *Funktionsmuster* für die Anlagenfunktion *Not-Halt* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Sicherheit

Mustername: Steuerung eines Not-Halts

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Not-Halt* zu erfüllen. In diesem Entwurfsmuster ist lediglich die Automatisierungsfunktion Not-Halt-Steuerung enthalten, die das Verhalten im Falle eines Not-Halts festlegt, die betreffenden Ressourcen ansteuert und das Erreichen eines sicheren Zustands ermöglicht.

Kontext: Dieses Entwurfsmuster findet in der Fertigungs- und in der Verfahrenstechnik Anwendung. Verwenden Sie dieses Funktionsmuster, wenn

- Gefahrenquellen im Automatisierungssystem auftreten und somit ein sofortiges Abschalten von Ressourcen benötigt wird.

Lösung: Die nachfolgende Abbildung zeigt die einschlägige Automatisierungsfunktion dieses Funktionsmusters.

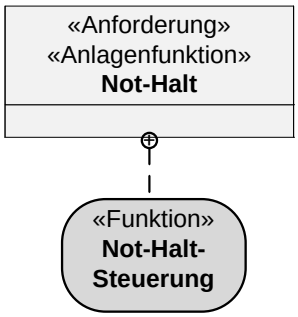

Abbildung C.2: Funktionsmuster »Not-Halt«

Teilnehmer: Dieses Funktionsmuster besteht aus einer Automatisierungsfunktion, die im Nachfolgenden näher beschrieben wird.

1. *Not-Halt-Steuerung:* Das Verhalten der Anlage bei Not-Halt (Stillsetzen im Notfall) und die direkte Ansteuerung der jeweils benötigten Ressourcen werden festgelegt. Hierbei sind die unterschiedlichen Not-Halt-Verfahren zu beachten:
   - stoppen (stillsetzen) der Anlage beziehungsweise Teilanlage und der Maschinen-Antriebseinheiten durch eine sofortige Unterbrechung der Energiezuführung (Stoppkategorie 0) oder
   - ein gesteuertes Stoppen (Stillsetzen), bei dem die Energiezufuhr zu den Antriebseinheiten der Anlage solange beibehalten wird, bis der Stillstand erzielt wurde (Stoppkategorie 1) – höchste Priorität für Not-Halt-Funktionen gegenüber allen anderen Funktionalitäten [vgl. ISO 13 850 2014, S. 7][#]

Anhang C 1.3 Funktionsmuster »Sortieren«

Im Nachfolgenden ist das *Funktionsmuster* der Anlagenfunktion *Sortieren* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Sortierung

Mustername: Sortieren von Komponenten inklusive Komponentenerkennung und Speicherung

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Sortieren* zu erfüllen. Die Automatisierungsfunktionen für die Komponentenerkennung – Komponente detektieren – und für die Speicherung eines Zahlenwerts – Komponente zählen und Zahlenwert archivieren – sind in diesem Entwurfsmuster enthalten.

Kontext: Anwendung findet dieses Entwurfsmuster hauptsächlich in der Fertigungstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- unterschiedlichen Komponenten verschiedene Transportwege zur Verfügung stehen.
- erfasst werden soll, wie oft eine Komponente in einem bestimmten Transportweg transportiert wurde.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

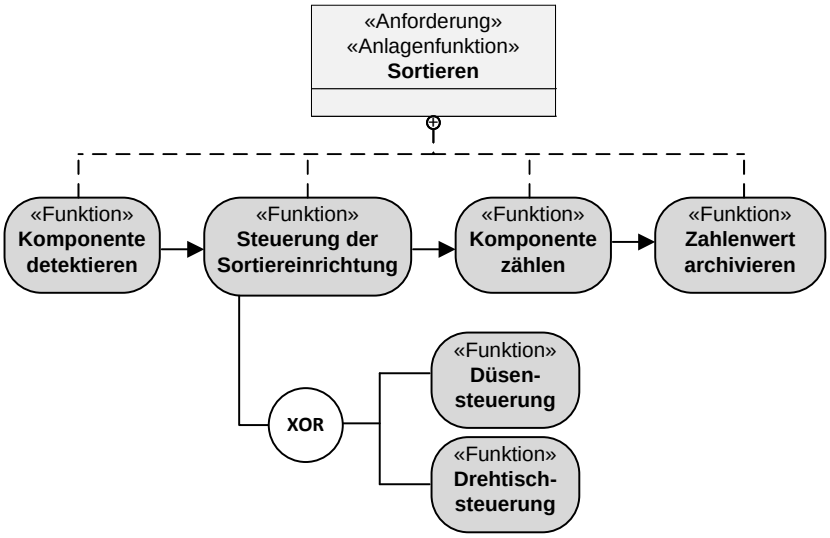

Abbildung C.3: Funktionsmuster »Sortieren«

Teilnehmer: Dieses Funktionsmuster besteht aus vier Automatisierungsfunktionen, die nachfolgend näher beschrieben werden.

1. *Komponente detektieren:* Direktes Erkennen beziehungsweise Identifizieren von Komponenten, wie beispielsweise Werkstücke
2. *Steuerung der Sortiereinrichtung:* Nachdem eine Komponente identifiziert wurde, muss die Sortiereinrichtung angesteuert werden, um diese Komponente auf den dafür vorgesehenen Transportweg weiterzuleiten.
3. *Komponente zählen:* Nachdem eine Komponente detektiert und an den vorgesehenen Transportweg weitergeleitet wurde, wird erfasst, wie oft dieser infrage stehende Transportweg bereits von Komponenten für den Weitertransport verwendet wurde.
4. *Zahlenwert archivieren:* Der erfasste Zahlenwert wird in dieser Automatisierungsfunktion gespeichert.

Anhang C 1.4 Funktionsmuster »Temperatur überwachen«

Nachfolgend ist das *Funktionsmuster* der Anlagenfunktion *Temperatur überwachen* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Temperatur

Mustername: Überwachen einer Temperatur inklusive messen und abgleichen

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Temperatur überwachen* zu erfüllen. Die Automatisierungsfunktionen für das Messen – Messgröße erfassen, Messgröße umwandeln und Messgröße filtern – und den Abgleich der Temperatur – Messgröße (Temperatur) abgleichen – sind in diesem Entwurfsmuster enthalten. Dieses Entwurfsmuster unterstützt die PLT-Stellen »T« und »TC«.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- die Temperatur einer bestimmten Komponente überwacht und mit einem Sollwert abgeglichen werden soll.
- auf eine zu hohe oder zu niedrige Temperatur reagiert werden soll.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

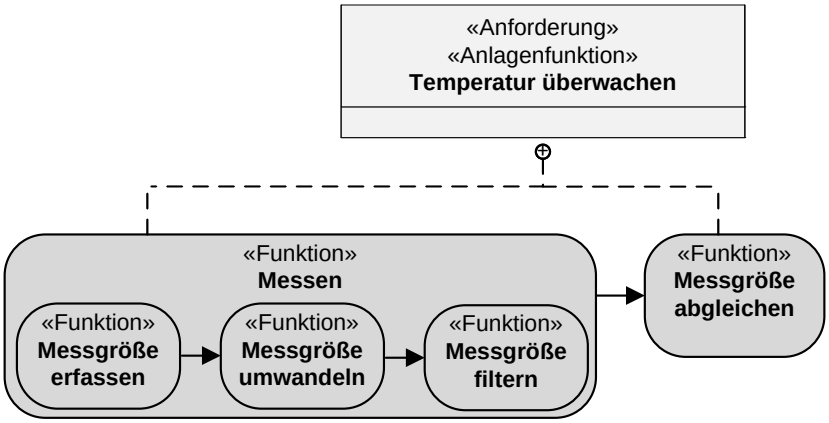

Abbildung C.4: Funktionsmuster »Temperatur überwachen«

Teilnehmer: Dieses Funktionsmuster besteht aus zwei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden.

1. *Messen:* Diese Automatisierungsfunktion besteht grundsätzlich aus mehreren untergliederten Automatisierungsfunktionen – Messgröße erfassen, Messgröße umwandeln und Messgröße filtern. In diesen Automatisierungsfunktionen wird die Messgröße zum einen erfasst sowie in ein elektrisches Signal umgewandelt und zum anderen gefiltert.
2. *Messgröße abgleichen:* Nach erfolgreichem Messen wird die gemessene Größe mit einem Sollwert verglichen, um so auf zu hohe oder zu niedrige Messwerte reagieren zu können.

Anhang C 1.5 Funktionsmuster »Hupen«

Nachfolgend ist das *Funktionsmuster* der Anlagenfunktion *Hupen* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Sicherheit

Mustername: Steuerung der Hupe inklusive Taktvorgabe

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Hupen* zu erfüllen. Die Automatisierungsfunktionen für die Taktvorgabe – Takt vorgeben und Takt senden – und Ansteuerung der Ressource – Steuerung der Signaleinrichtung – sind in diesem Entwurfsmuster enthalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- eine Hupenwarnung erzeugt und hierzu ein Takt vorgegeben werden soll.
- auf mögliche Fehler mithilfe eines Hupensignals aufmerksam gemacht werden soll.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

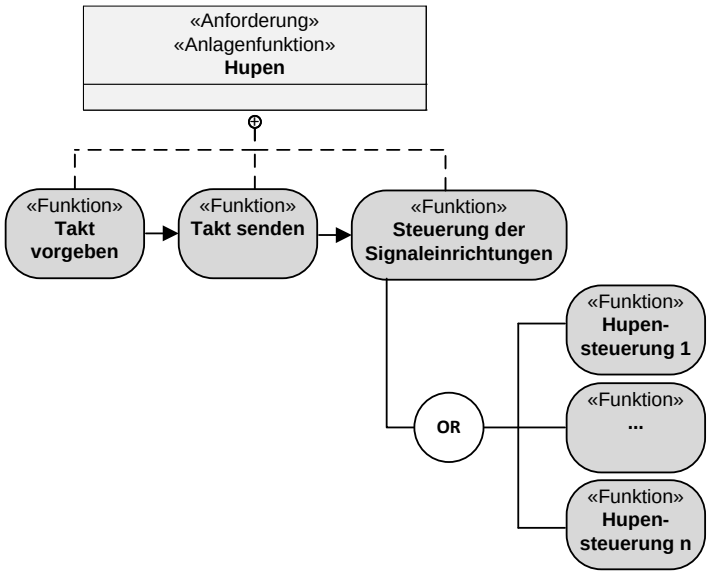

Abbildung C.5: Funktionsmuster »Hupen«

Teilnehmer: Dieses Funktionsmuster besteht aus drei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden.

1. *Takt vorgeben:* Je nach Problemstellung beziehungsweise Status der Anlage werden möglicherweise unterschiedliche Hupensignale benötigt. In dieser Automatisierungsfunktion wird der Takt festgelegt.
2. *Takt senden:* Nachdem der Takt festgelegt wurde, muss dieser den benötigten Ressourcen zur Verfügung gestellt werden.
3. *Steuerung der Signaleinrichtung:* Direktes Ansteuern der Hupe. Das Hupensignal wird durch die Taktbereitstellung vorgegeben.

Anhang C 1.6 Funktionsmuster »Füllstand protokollieren«

Im Folgenden ist das *Funktionsmuster* der Anlagenfunktion *Füllstand protokollieren* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Füllstand

Mustername: Füllstand protokollieren inklusive messen, abgleichen und archivieren

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Füllstand protokollieren* zu erfüllen. Die Automatisierungsfunktionen für das Messen – Messgröße erfassen, Messgröße umwandeln und Messgröße filtern –, der Abgleich der Messgröße – Messgröße abgleichen – sowie die Archivierung – Messgröße archivieren – sind in diesem Entwurfsmuster enthalten. Dieses Entwurfsmuster unterstützt die PLT-Stellen »L«, »LC«, »LR« und alle Kombinationen.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- der Füllstand einer bestimmten Komponente erfasst, mit einem Sollwert abgeglichen und archiviert werden soll.
- auf einen zu hohen oder zu niedrigen Füllstand reagiert werden soll.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

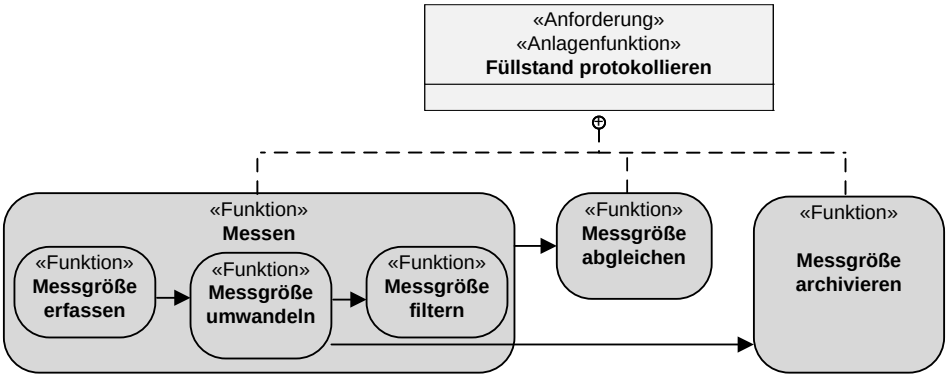

Abbildung C.6: Funktionsmuster »Füllstand protokollieren«

Teilnehmer: Dieses Funktionsmuster besteht aus drei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden.

1. *Messen:* Diese Automatisierungsfunktion besteht grundsätzlich aus mehreren untergliederten Automatisierungsfunktionen – Messgröße erfassen, Messgröße umwandeln und Messgröße filtern. In diesen Automatisierungsfunktionen wird die Messgröße zum einen erfasst sowie in ein elektrisches Signal umgewandelt und zum anderen gefiltert.
2. *Messgröße abgleichen:* Nach erfolgreichem Messen wird die gemessene Größe mit einem Sollwert verglichen, um so auf einen zu hohen oder zu niedrigen Füllstand reagieren zu können.
3. *Messgröße archivieren:* Nachdem die Messgröße umgewandelt wurde, wird diese gespeichert.

Anhang C 2 Verteilungsmuster

Im Folgenden sind die Verteilungsmuster für die fördertechnische Anlage (siehe Abschnitt 9.1) dargestellt.

Anhang C 2.1 Verteilungsmuster »Transportieren«

Im Nachfolgenden ist das *Verteilungsmuster Transportieren* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Komponenten transportieren

Mustername: Verteiltes Transportieren exklusiv Wegverfolgung und Reservierung

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Ressourcenansteuerung unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Transportmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- mehrere Knoten zur Verfügung stehen, um die Funktionalität auf diese zu verteilen und somit die Ressourcenansteuerung separat auf unterschiedlichen Knoten stattfindet.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Transportieren* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Transportwege sowie die Verteilung der dafür benötigten Funktionalität auf zwei Knoten.

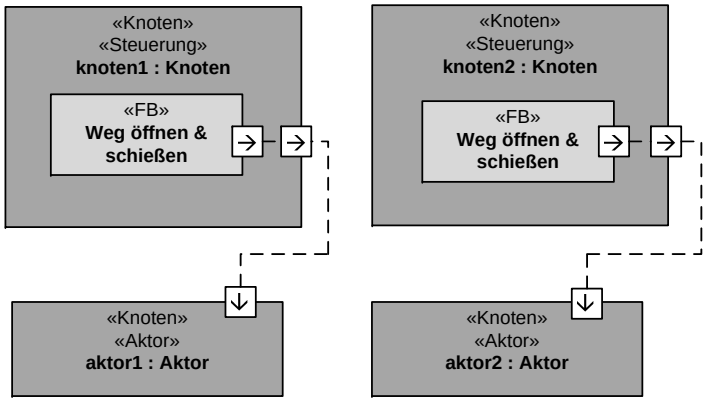

Abbildung C.7: Verteilungsmuster »Transportieren«

Teilnehmer: Dieses Verteilungsmuster besteht aus zwei Knoten sowie zwei Aktoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1 und Knoten 2:* Die Funktionalität der Ressourcenansteuerung wird für jede Ressource auf einem separaten Knoten realisiert.
2. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, die geöffnet beziehungsweise angeschaltet oder geschlossen beziehungsweise ausgeschaltet werden müssen.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle C.1: Anforderungsmerkmale im Verteilungsmuster »Transportieren«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle C.2: Lösungsmerkmale im Verteilungsmuster »Transportieren«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Ressourcennutzung: Es können leicht neue Transportwege hinzugefügt werden, ohne vorhandene Knoten anzupassen. Da jeder Transportweg auf einem separaten Knoten implementiert wird, können kostengünstige und kleine Knoten eingesetzt werden. Ist die Anzahl der Transportwege überschaubar gering, sollte dieses Entwurfsmuster eingesetzt werden. Änderungen an diesem Knoten sind leicht implementierbar, da die Knoten nicht sonderlich komplex sind, im Gegensatz dazu, wenn nur ein Knoten für alle Transportwege implementiert wird.
- Verfügbarkeit: Fällt ein Knoten aus, steht nur der Teil der Anlage still, für den der Knoten zuständig war und nicht die gesamte Anlage. Es können alternative Transportwege gesucht werden.

Nachteile: Dieses Entwurfsmuster hat einen Nachteil – siehe wie folgt.

- Verfügbarkeit: Fällt ein Knoten für einen Streckenabschnitt aus, kann auf diesen Streckenabschnitt nicht mehr zugegriffen werden.

Anhang C 2.2 Verteilungsmuster »Not-Halt«

Im Nachfolgenden ist das *Verteilungsmuster Not-Halt* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Not-Halt

Mustername: Verteilung der Not-Halt-Steuerung

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Not-Halt-Steuerung unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Sicherheitsmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- ein Knoten zur Verfügung steht, der die Not-Halt-Steuerung übernehmen soll.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Not-Halt* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Not-Halte sowie die Verteilung der dafür benötigten Funktionalität auf einen Knoten.

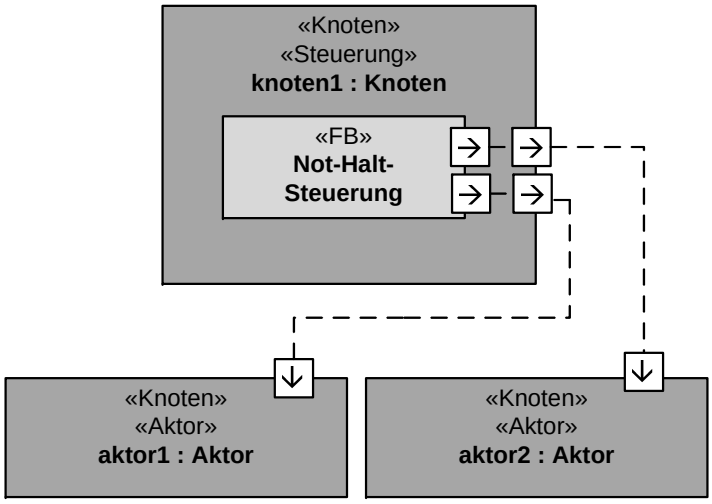

Abbildung C.8: Verteilungsmuster »Not-Halt«

Teilnehmer: Dieses Verteilungsmuster besteht aus einem Knoten sowie zwei Aktoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1:* Die Funktionalität der Not-Halt-Steuerung wird für alle Ressourcen auf einem zentralen Knoten realisiert.
2. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, die bei einem Not-Halt angesprochen werden müssen.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle C.3: Anforderungsmerkmale im Verteilungsmuster »Not-Halt«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle C.4: Lösungsmerkmale im Verteilungsmuster »Not-Halt«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
		Speicherkompatibilität (load)	
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
		Speicherkompatibilität (runtime)	
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
		Klemme-Klemme-Reaktionszeit	
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Zeitverhalten: Da in einem Notfall ein Not-Halt schnell durchgeführt werden muss und bei dieser Verteilungsvariante lediglich ein Knoten angesprochen wird, kann diese Verteilungsvariante schneller reagieren.

Nachteile: Dieses Entwurfsmuster hat einen Nachteil – siehe wie folgt.

- Verfügbarkeit: Fällt ein Knoten aus, kann die Not-Halt-Funktionalität für die gesamte Anlage nicht mehr realisiert werden.

Anhang C 2.3 Verteilungsmuster »Sortieren«

Im Folgenden ist das *Verteilungsmuster Sortieren* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Komponenten sortieren

Mustername: Verteiltes Sortieren inklusive Komponentenerkennung und Speicherung

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Sortierung unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Transportmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- ein Knoten zur Verfügung steht, der mehrere Sortiereinrichtungen steuert.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Sortieren* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Sortiereinrichtungen sowie die Verteilung der dafür benötigten Funktionalität auf einen Knoten.

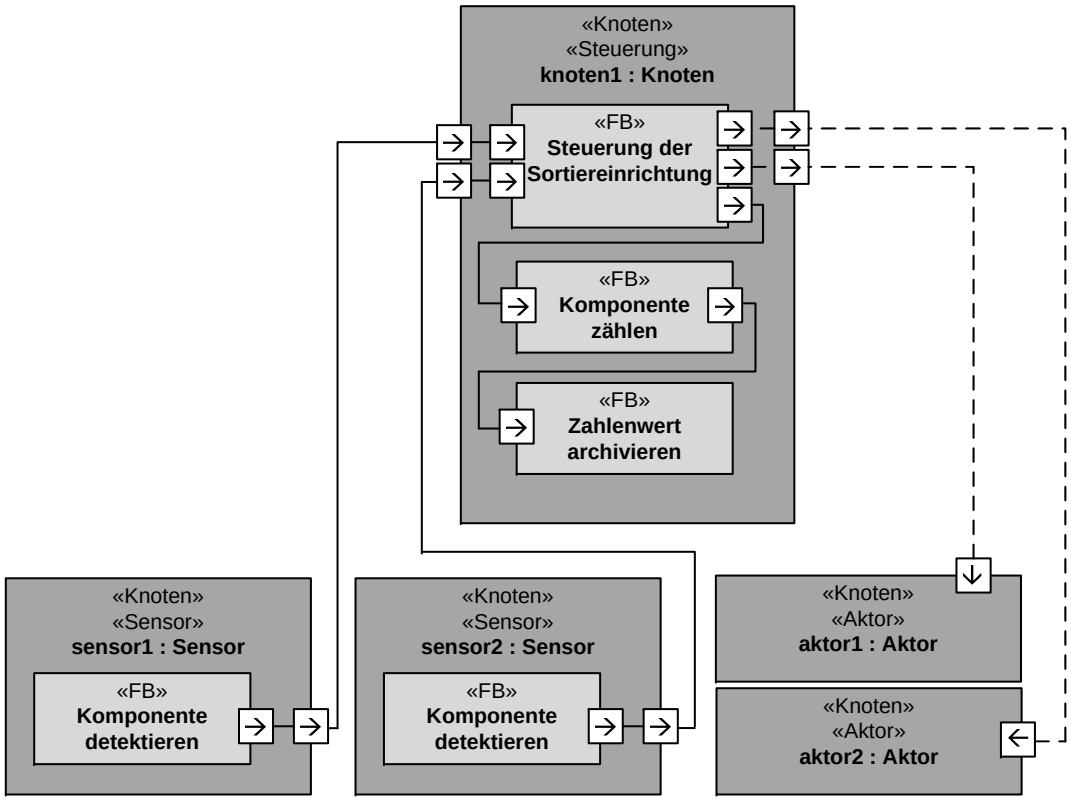

Abbildung C.9: Verteilungsmuster »Sortieren«

Teilnehmer: Dieses Verteilungsmuster besteht aus einem Knoten sowie zwei Aktoren und zwei Sensoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1:* Die Funktionalität der Ressourcenansteuerung, Komponente zählen und Zahlwert erfassen wird für jede Ressource auf einem zentralen Knoten realisiert.
2. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, welche die Sortierung übernehmen.

3. *Sensor 1 und Sensor 2:* Diese Sensoren erkennen die Komponente und sind somit für deren Detektion zuständig. Sobald eine Komponente erkannt wird, erfolgt die Ansteuerung der Sortiereinrichtung.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle C.5: Anforderungsmerkmale im Verteilungsmuster »Sortieren«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle C.6: Lösungsmerkmale im Verteilungsmuster »Sortieren«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Ressourcennutzung: Findet die Steuerung der Sortiereinrichtung und die Speicherung der Anzahl der Komponenten auf nur einem Knoten statt, müssen die anderen Knoten keine Ressourcen für eine Speicherung zur Verfügung stellen. Dies hat zur Folge, dass Kosten reduziert werden. In diesem Entwurfsmuster ist lediglich ein Archivierungsknoten notwendig, der kostengünstiger ist, als mehrere kleine lokale Archivierungsknoten. Es können leichter neue zu steuernde Sortiereinrichtungen angeschlossen werden, ohne neue Knoten hinzufügen zu müssen.
- Verfügbarkeit: Fällt der Knoten, der für das Erkennen von Komponenten zuständig ist, aus, kann trotzdem noch auf die archivierte Anzahl von Komponenten zugegriffen werden.

Nachteile: Dieses Entwurfsmuster besitzt unterschiedliche Nachteile, die im Folgenden aufgelistet werden.

- Verfügbarkeit: Fällt der Knoten aus, auf dem die Messgrößen archiviert wurden, kann auf die archivierten Daten nicht mehr zugegriffen werden. Dies bedeutet, dass ein Remote-Zugriff auf das Archiv beziehungsweise die archivierten Daten nicht mehr möglich ist. Der Ausfall hätte einen größeren Verlust zur Folge als bei einer »vor Ort«-Speicherung, da auf mehrere Messgrößen nicht mehr zugegriffen werden kann. Des Weiteren können bei Ausfall eines Sensors aus der Gruppe der Sensoren, die für das Detektieren zuständig ist, die Komponenten nicht mehr sortiert werden, da nicht garantiert werden kann, dass der zweite Sensor die Funktionalität des ausgefallenen Sensors übernimmt.
- Ressourcennutzung: Wird eine neue Sortiereinrichtung hinzugefügt, muss der Knoten angepasst werden.

Anhang C 2.4 Verteilungsmuster »Messgröße überwachen«

Im Nachfolgenden ist das *Verteilungsmuster Messgröße überwachen* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Messgröße überwachen

Mustername: Verteiltes Überwachen der Messgröße inklusive messen und abgleichen

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Überwachung einer Messgröße unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Verteilungsmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- mehrere Knoten zur Verfügung stehen, die jeweils eine Überwachung übernehmen.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Messgröße überwachen* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Überwachungen sowie die Verteilung der dafür benötigten Funktionalität auf zwei Knoten.

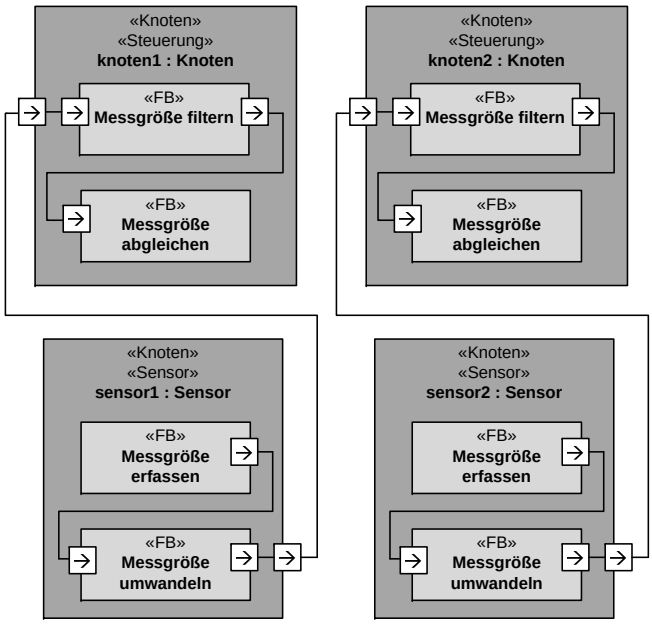

Abbildung C.10: Verteilungsmuster »Messgröße überwachen«

Teilnehmer: Dieses Verteilungsmuster besteht aus zwei Knoten sowie zwei Sensoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1 und Knoten 2:* Die Funktionalität der Messgrößenfilterung sowie -abgleichung findet für jede Messwertüberwachung auf einem separaten Knoten statt.
2. *Sensor 1 und Sensor 2:* Diese Sensoren sind jeweils für das Erfassen der Messgröße und für das Umwandeln in ein elektrisches Signal zuständig.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle C.7: Anforderungsmerkmale im Verteilungsmuster »Messgröße überwachen«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle C.8: Lösungsmerkmale im Verteilungsmuster »Messgröße überwachen«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Verfügbarkeit: Fällt ein Knoten aus, können die anderen Knoten die Messwertüberwachung fortsetzen und bei Bedarf die Funktionalität des ausgefallenen Knotens übernehmen.
- Ressourcennutzung: Es können leicht neue Messwertüberwachungen hinzugefügt werden, ohne vorhandene Knoten anzupassen. Es wird eine relativ hohe Anzahl an Knoten benötigt, die kostengünstig sein können.

Nachteile: Dieses Entwurfsmuster hat einen Nachteil – siehe wie folgt.

- Verfügbarkeit: Fällt ein Knoten für die Messwertüberwachung aus, muss ein anderer diese Funktionalität übernehmen, oder es ist keine Überwachung mehr möglich.

Anhang C 2.4 Verteilungsmuster »Hupen«

Im Nachfolgenden ist das *Verteilungsmuster Hupen* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Hupen

Mustername: Verteilte Ansteuerung der Hupe inklusive Taktvorgabe

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Ansteuerung der Hupe unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Verteilungsmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- mehrere Knoten zur Verfügung stehen, die jeweils eine Hupenansteuerung übernehmen.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Hupen* auf zwei Knoten verteilt. Die nachfolgende Lösung zeigt zwei Hupenansteuerungen sowie die Verteilung der dafür benötigten Funktionalität auf drei Knoten.

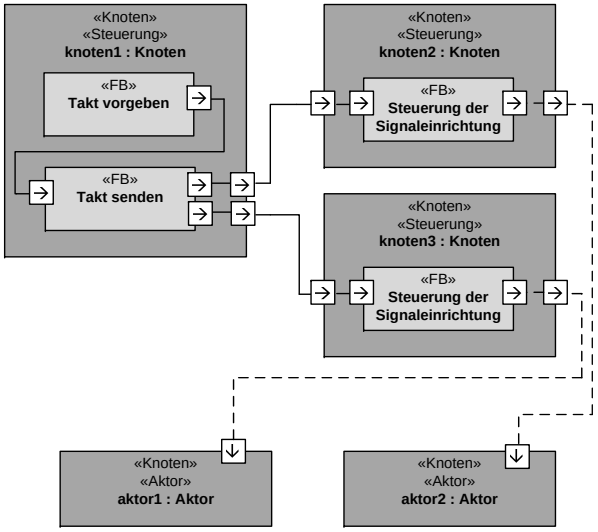

Abbildung C.11: Verteilungsmuster »Hupen«

Teilnehmer: Dieses Verteilungsmuster besteht aus drei Knoten sowie zwei Aktoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1:* Die Funktionalität der Taktvorgabe findet für alle anzusteuernden Hupen auf einem zentralen Knoten statt.
2. *Knoten 2 und Knoten 3:* Die Funktionalität der Hupenansteuerung findet für jede Hupe auf einem separaten Knoten statt.
3. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, die an- und ausgeschaltet werden müssen.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle C.9: Anforderungsmerkmale im Verteilungsmuster »Hupen«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle C.10: Lösungsmerkmale im Verteilungsmuster »Hupen«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
		Speicherkompatibilität (load)	
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
		Speicherkompatibilität (runtime)	
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
		Klemme-Klemme-Reaktionszeit	
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Verfügbarkeit: Fällt der Knoten aus, auf dem die Sicherheitseinrichtung gesteuert wird, ist nur der Teil der Anlage betroffen, für den der Knoten zuständig ist und nicht die gesamte Anlage.
- Ressourcennutzung: Es können neue Sicherheitseinrichtungen hinzugefügt werden, ohne vorhandene Knoten anzupassen. Es wird eine hohe Anzahl an Knoten benötigt, die kostengünstig sein können.

Nachteile: Dieses Entwurfsmuster hat einen Nachteil – siehe wie folgt.

- Verfügbarkeit: Fällt ein Knoten für die Ansteuerung der Sicherheitseinrichtung aus, kann diese nicht mehr angesteuert werden. Diese Funktionalität müsste dann von einem anderen Knoten übernommen werden.

Anhang C 2.5 Verteilungsmuster »Messgröße protokollieren«

Im Folgenden ist das *Verteilungsmuster Messgröße protokollieren* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Messgröße protokollieren

Mustername: Verteiltes Protokollieren der Messgröße inklusive messen, abgleichen und archivieren

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Messgrößenerfassung, -abgleichung und -archivierung unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Verteilungsmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- ein Knoten die Messwertabgleichung für alle zu erfassenden Messgrößen übernehmen soll.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Messgröße protokollieren* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Überwachungen sowie die Verteilung der dafür benötigten Funktionalität auf einen Knoten.

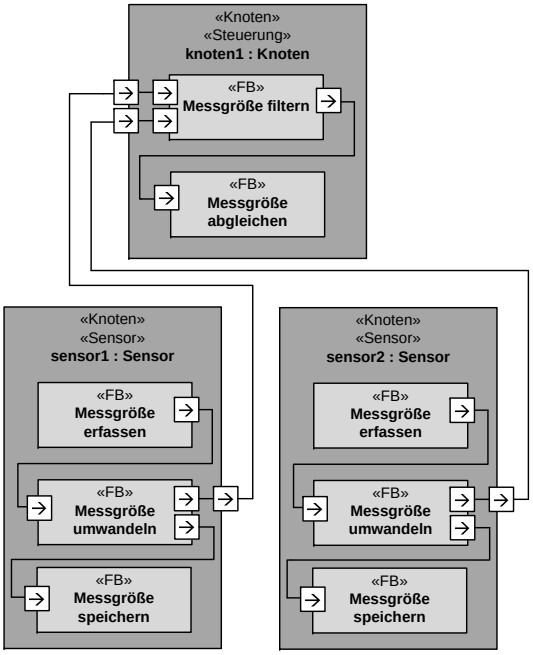

Abbildung C.12: Verteilungsmuster »Messgröße protokollieren«

Teilnehmer: Dieses Verteilungsmuster besteht aus einem Knoten sowie zwei Sensoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1:* Dieser Knoten übernimmt die Messgrößenfilterung und -abgleichung für alle zu erfassenden Messgrößen.
2. *Sensor 1 und Sensor 2:* Diese Sensoren sind jeweils für das Erfassen und Umwandeln der relevanten Messgröße zuständig. Des Weiteren wird der Messwert jeweils vor Ort auf dem jeweiligen Sensor gespeichert.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle C.11: Anforderungsmerkmale im Verteilungsmuster »Messgröße protokollieren«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle C.12: Lösungsmerkmale im Verteilungsmuster »Messgröße protokollieren«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Ressourcennutzung: Findet die Speicherung der Messgröße auf dem Sensor statt, müssen die anderen Knoten keine Ressourcen für eine Speicherung zur Verfügung stellen. Dies hat zur Folge, dass Kosten reduziert werden. In diesem Entwurfsmuster müssen lediglich die Sensoren Archivierungsressourcen zur Verfügung stellen. Es können leichter neue zu erfassende Messgrößen hinzugefügt werden, ohne bestehende Knoten anpassen zu müssen.
- Verfügbarkeit: Fällt der Knoten, der für die Abgleichung und Filterung der Messgröße zuständig ist, aus, kann trotzdem noch auf die archivierten Daten zugegriffen werden.

Nachteile: Dieses Entwurfsmuster besitzt unterschiedliche Nachteile, die im Folgenden aufgelistet werden.

- Ressourcennutzung: Jeder Sensor benötigt Ressourcen für die Speicherung, was bei einer hohen Anzahl an Sensoren teuer ist.
- Verfügbarkeit: Fällt ein Sensor aus, ist ein Remote-Zugriff auf das Archiv beziehungsweise auf die Archivdaten nicht mehr möglich.

Anhang D Entwurfsmuster der verfahrenstechnischen Anlage

Anhang D 1 Funktionsmuster

Im Folgenden sind die Funktionsmuster für die verfahrenstechnische Anlage (siehe Abschnitt 9.2) dargestellt.

Anhang D 1.1 Funktionsmuster »Transportieren«

Im Nachfolgenden ist das *Funktionsmuster* für die Anlagenfunktion *Transportieren* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Transport

Mustername: Transportieren von Komponenten exklusiv Wegverfolgung und Reservierung

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Transportieren* zu erfüllen. In diesem Entwurfsmuster wird lediglich die Ressourcenansteuerung betrachtet. Die Automatisierungsfunktionen für die Wegverfolgung – Weg suchen und Weg wählen – und für die Reservierung – Weg reservieren und Reservierung aufheben – sind in diesem Entwurfsmuster nicht enthalten.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- für den Transport nicht mehrere Transportwege zur Verfügung stehen.
- das Suchen und Reservieren von Ressourcen nicht benötigt wird, weil auf diese nicht gleichzeitig zugegriffen wird.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

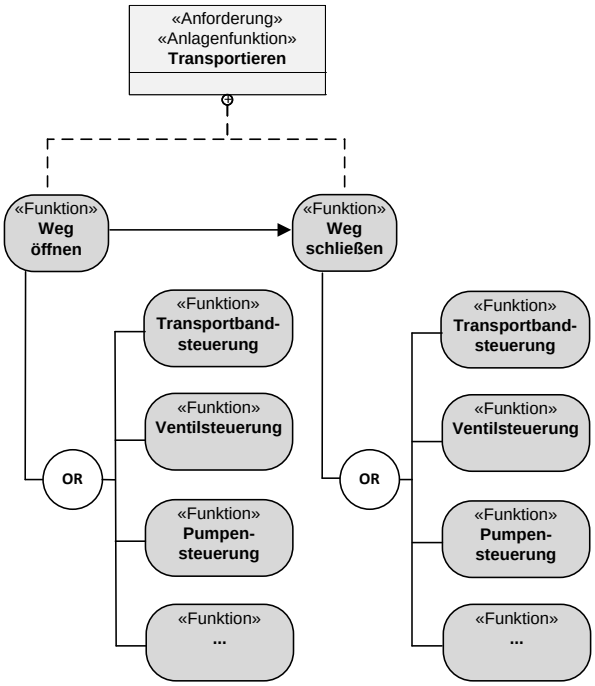

Abbildung D.1: Funktionsmuster »Transportieren ohne Reservierung«

Teilnehmer: Dieses Funktionsmuster besteht aus zwei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden.

1. *Weg öffnen:* Direktes Ansteuern der benötigten Ressourcen ohne zusätzliche Reservierung. Dies bedeutet, dass der Weg bei Bedarf für den Transport geöffnet wird. Zum Öffnen von Ressourcen, wie beispielsweise Anschalten von Förderbändern oder Öffnen von Ventilen, stehen mehrere Verfahren zur Verfügung. Hierbei ist zu beachten, dass die jeweiligen Ressourcen zum einen lediglich angeschaltet beziehungsweise geöffnet und zum anderen mit bestimmten Parametern, wie beispielsweise Geschwindigkeit, angeschaltet beziehungsweise geöffnet werden können.
2. *Weg schließen:* Nach erfolgreichem Transportieren der Komponenten muss der Transportweg geschlossen werden. Dies bedeutet, dass beispielsweise Förderbänder ausgeschaltet und Ventile geschlossen werden. Hierzu stehen die gleichen Verfahren wie beim Weg öffnen zur Verfügung.

Anhang D 1.2 Funktionsmuster »Füllstand überwachen«

Im Nachfolgenden ist das *Funktionsmuster* der Anlagenfunktion *Füllstand überwachen* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Füllstand

Mustername: Überwachen eines Füllstands inklusive messen und abgleichen

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Füllstand überwachen* zu erfüllen. Die Automatisierungsfunktionen für das Messen – Messgröße erfassen, Messgröße umwandeln und Messgröße filtern – und den Abgleich des Füllstands – Messgröße abgleichen – sind in diesem Entwurfsmuster enthalten. Dieses Entwurfsmuster unterstützt die PLT-Stellen »L« und »LC«.

Kontext: Anwendung findet dieses Entwurfsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- der Füllstand einer bestimmten Komponente überwacht und mit einem Sollwert abgeglichen werden soll.
- auf einen zu hohen oder zu niedrigen Füllstand reagiert werden soll.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

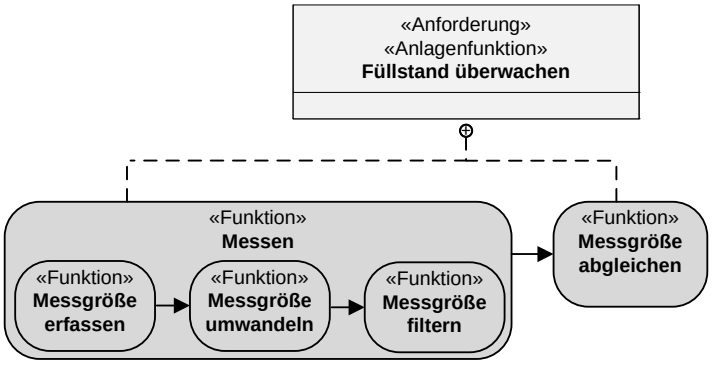

Abbildung D.2: Funktionsmuster »Füllstand überwachen«

Teilnehmer: Dieses Funktionsmuster besteht aus zwei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden

1. *Messen:* Diese Automatisierungsfunktion besteht grundsätzlich aus mehreren untergliederten Automatisierungsfunktionen – Messgröße erfassen, Messgröße umwandeln und Messgröße filtern. In diesen Automatisierungsfunktionen wird die Messgröße zum einen erfasst sowie in ein elektrisches Signal umgewandelt und zum anderen gefiltert.
2. *Messgröße abgleichen:* Nach erfolgreichem Messen wird die gemessene Größe mit einem Sollwert verglichen, um so auf zu hohe oder zu niedrige Messwerte reagieren zu können. Der Sollwert wird als lokale Variable im Funktionsblock gespeichert.

Anhang D 1.3 Funktionsmuster »Mischen«

Im Nachfolgenden ist das *Funktionsmuster* der Anlagenfunktion *Mischen* dargestellt.

Musterkategorie: Funktionsmuster

Mustertyp: Bearbeitung

Mustername: Mischen von Komponenten inklusive Dosierung

Zweck: Dieses Entwurfsmuster stellt sicher, dass alle relevanten Automatisierungsfunktionen beachtet werden, um die Anlagenfunktion *Mischen* zu erfüllen. Die Automatisierungsfunktion für die Dosierung – Flüssigkeiten dosieren – ist in diesem Entwurfsmuster enthalten.

Kontext: Anwendung findet dieses Funktionsmuster hauptsächlich in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- eine beliebige Anzahl an Flüssigkeiten dosiert werden muss.
- unterschiedliche Komponenten miteinander gemischt werden sollen.

Lösung: Die nachfolgende Abbildung zeigt die einschlägigen Automatisierungsfunktionen dieses Funktionsmusters sowie deren Interaktionen.

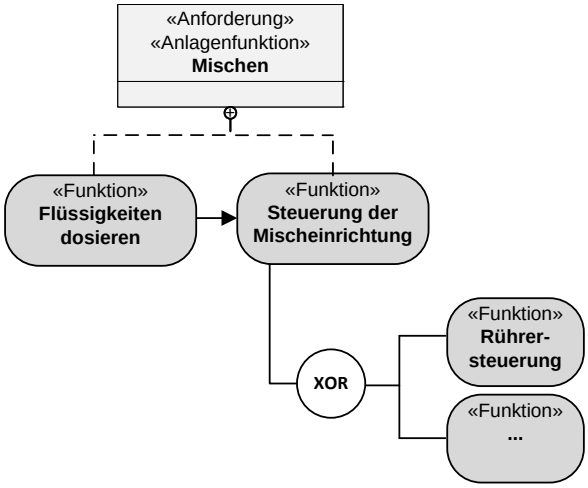

Abbildung D.3: Funktionsmuster »Mischen«

Teilnehmer: Dieses Funktionsmuster besteht aus zwei Automatisierungsfunktionen, die im Nachfolgenden näher beschrieben werden.

1. *Flüssigkeiten dosieren:* In dieser Automatisierungsfunktion wird eine beliebige Anzahl Flüssigkeiten dosiert. Hierbei ist zu beachten, dass eine Mengenangabe pro Flüssigkeit benötigt wird. Des Weiteren muss festgelegt werden, ob die Flüssigkeiten nacheinander dosiert werden oder gleichzeitig.
2. *Steuerung der Mischeinrichtung:* Nachdem die Flüssigkeiten dosiert wurden, muss die Mischeinrichtung angesteuert werden, um die Flüssigkeit(en) zu mischen. Hierbei ist die Ansteuerung der Mischeinrichtung zu beachten. Eine Alternative ist es, die Mischeinrichtung erst dann einzuschalten, wenn *alle* Flüssigkeiten dosiert wurden oder aber alternativ sie sofort einzuschalten, sobald *eine* Flüssigkeit dosiert wurde. Eine weitere Möglichkeit ist das Mischen mit Pausen. Zur Ansteuerung der Mischeinrichtung stehen mehrere Verfahren zur Verfügung. Hierbei ist zu beachten, dass die jeweiligen Ressourcen zum einen lediglich angeschaltet und zum anderen mit bestimmten Parametern, wie beispielsweise Geschwindigkeit, angeschaltet werden können.

Anhang D 2 Verteilungsmuster

Im Folgenden sind die Verteilungsmuster für die verfahrenstechnische Anlage (siehe Abschnitt 9.2) dargestellt.

Anhang D 2.1 Verteilungsmuster »Transportieren«

Im Nachfolgenden ist das *Verteilungsmuster Transportieren* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Komponenten transportieren

Mustername: Verteiltes Transportieren exklusiv Wegverfolgung und Reservierung

Zweck: Dieses Verteilungsmuster unterstützt die verteilte Ausführung der Ressourcenansteuerung unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Verteilungsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Transportmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- mehrere Knoten zur Verfügung stehen, um die Funktionalität auf diese zu verteilen und somit die Ressourcenansteuerung separat auf unterschiedlichen Knoten stattfindet.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Transportieren* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Transportwege sowie die Verteilung der dafür benötigten Funktionalität auf zwei Knoten.

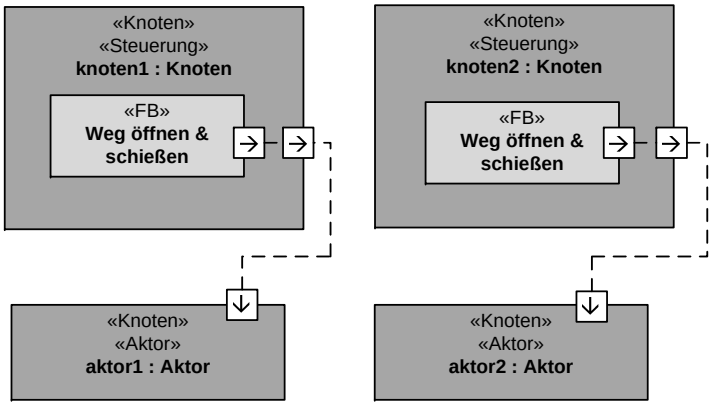

Abbildung D.4: Verteilungsmuster »Transportieren«

Teilnehmer: Dieses Verteilungsmuster besteht aus zwei Knoten sowie zwei Aktoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1 und Knoten 2:* Die Funktionalität der Ressourcenansteuerung wird für jede Ressource auf einem separaten Knoten realisiert.
2. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, die geöffnet beziehungsweise angeschaltet oder geschlossen beziehungsweise ausgeschaltet werden müssen.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle D.1: Anforderungsmerkmale im Verteilungsmuster »Transportieren«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle D.2: Lösungsmerkmale im Verteilungsmuster »Transportieren«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Ressourcennutzung: Es können leicht neue Transportwege hinzugefügt werden, ohne vorhandene Knoten anzupassen. Da jeder Transportweg auf einem separaten Knoten implementiert wird, können kostengünstige und kleine Knoten eingesetzt werden. Ist die Anzahl der Transportwege überschaubar gering, sollte dieses Entwurfsmuster eingesetzt werden. Änderungen an diesem Knoten sind leicht implementierbar, da die Knoten nicht sonderlich komplex sind, im Gegensatz dazu, wenn nur ein Knoten für alle Transportwege implementiert wird.
- Verfügbarkeit: Fällt ein Knoten aus, steht nur der Teil der Anlage still, für den der Knoten zuständig war und nicht die gesamte Anlage. Es können alternative Transportwege gesucht werden.

Nachteile: Dieses Entwurfsmuster hat einen Nachteil – siehe wie folgt.

- Verfügbarkeit: Fällt ein Knoten für einen Streckenabschnitt aus, kann auf diesen Streckenabschnitt nicht mehr zugegriffen werden.

Anhang D 2.2 Verteilungsmuster »Messgröße überwachen«

Im Nachfolgenden ist das *Verteilungsmuster Messgröße überwachen* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Messgröße überwachen

Mustername: Verteiltes Überwachen der Messgröße inklusive messen und abgleichen

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung der Überwachung einer Messgröße unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Verteilungsmuster in der Fertigungs- und in der Verfahrenstechnik. Verwenden Sie dieses Verteilungsmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- mehrere Knoten zur Verfügung stehen, die jeweils eine Überwachung übernehmen.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *Messgröße überwachen* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Überwachungen sowie die Verteilung der dafür benötigten Funktionalität auf zwei Knoten.

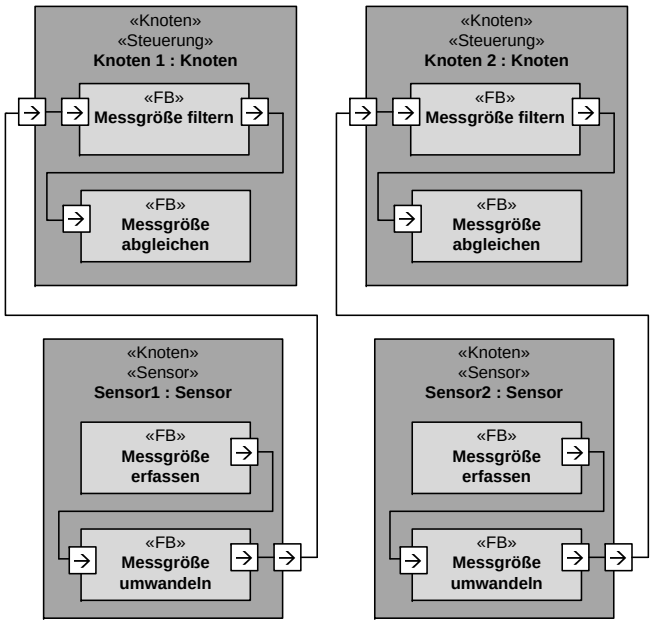

Abbildung D.5: Verteilungsmuster »Messgröße überwachen«

Teilnehmer: Dieses Verteilungsmuster besteht aus zwei Knoten sowie zwei Sensoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1 und Knoten 2:* Die Funktionalität der Messgrößenfilterung sowie -abgleichung findet für jede Messwertüberwachung auf einem separaten Knoten statt.
2. *Sensor 1 und Sensor 2:* Diese Sensoren sind jeweils für das Erfassen der Messgröße und für das Umwandeln in ein elektrisches Signal zuständig.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle D.3: Anforderungsmerkmale im Verteilungsmuster »Messgröße überwachen«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle D.4: Lösungsmerkmale im Verteilungsmuster »Messgröße überwachen«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Verfügbarkeit: Fällt ein Knoten aus, können die anderen Knoten die Messwertüberwachung fortsetzen und bei Bedarf die Funktionalität des ausgefallenen Knotens übernehmen.
- Ressourcennutzung: Es können leicht neue Messwertüberwachungen hinzugefügt werden, ohne vorhandene Knoten anzupassen. Es wird eine relativ hohe Anzahl an Knoten benötigt, die kostengünstig sein können.

Nachteile: Dieses Entwurfsmuster hat einen Nachteil – siehe wie folgt.

- Verfügbarkeit: Fällt ein Knoten für die Messwertüberwachung aus, muss ein anderer diese Funktionalität übernehmen oder es ist keine Überwachung mehr möglich.

Anhang D 2.3 Verteilungsmuster »Mischen«

Im Nachfolgenden ist das *Verteilungsmuster Mischen* dargestellt.

Musterkategorie: Verteilungsmuster

Mustertyp: Komponenten mischen

Mustername: Verteiltes Mischen inklusive Dosierung

Zweck: Dieses Entwurfsmuster unterstützt die verteilte Ausführung des Mischens unter Berücksichtigung der nfAs Ressourcennutzung und Zeitverhalten.

Kontext: Anwendung findet dieses Verteilungsmuster hauptsächlich in der Verfahrenstechnik. Verwenden Sie dieses Funktionsmuster, wenn

- die nfAs Ressourcennutzung und Zeitverhalten eine wichtige Rolle spielen.
- ein Knoten zur Verfügung steht, der mehrere Mischeinrichtungen steuert.

Lösung: Die nachfolgende Abbildung zeigt die Verteilung der einschlägigen Automatisierungsfunktionen auf Knoten sowie deren Interaktionen. Gemäß dieser Lösung wird eine Anlagenfunktion *verteiltes Mischen* auf einen Knoten verteilt. Die nachfolgende Lösung zeigt zwei Sortiereinrichtungen sowie die Verteilung der dafür benötigten Funktionalität auf einen Knoten.

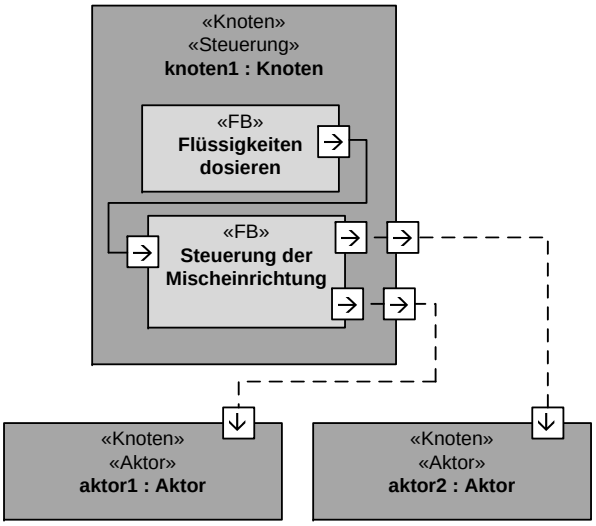

Abbildung D.6: Verteilungsmuster »Mischen«

Teilnehmer: Dieses Verteilungsmuster besteht aus einem Knoten sowie zwei Aktoren, die im Nachfolgenden näher beschrieben werden.

1. *Knoten 1:* Die Funktionalität Flüssigkeit dosieren sowie Steuerung der Mischeinrichtung wird für jede Ressource auf einem zentralen Knoten realisiert.
2. *Aktor 1 und Aktor 2:* Die Aktoren sind die anzusteuernden Ressourcen, die das Mischen übernehmen.

Konsequenzen: In diesem Abschnitt werden die relevanten Anforderungs- und Lösungsmerkmale beschrieben. Das Ausfüllen beziehungsweise Berechnen dieser Merkmale ist erst bei einer kontextbezogenen Anwendung des Verteilungsmusters möglich.

Tabelle D.5: Anforderungsmerkmale im Verteilungsmuster »Mischen«

Anforderungsmerkmale	
Merkmalsträger	Ausprägung
Steuerungszykluszeit	
Knoten	Diese Daten werden automatisch ausgefüllt.
Gerät-Durchlaufzeit	
Gerät	Diese Daten werden automatisch ausgefüllt.
Ausführungszeit einer POU	
Funktion	Diese Daten werden automatisch ausgefüllt.
Feldbuszykluszeit	
Funktion	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Runtime)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Bereitgestellter Speicher (Load)	
Knoten	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Runtime)	
Funktion	Diese Daten werden automatisch ausgefüllt.
Benötigter Speicher (Load)	
Funktion	Diese Daten werden automatisch ausgefüllt.

Tabelle D.6: Lösungsmerkmale im Verteilungsmuster »Mischen«

Lösungsmerkmale			
Trend	Merkmalsträger	Berechnung	Ausprägung
Speicherkompatibilität (load)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Speicherkompatibilität (runtime)			
✔	Knoten	Bereitgestellter Speicher ↔ Benötigter Speicher	`IF` Bereitgestellter Speicher < Benötigter Speicher `THEN` false `ELSE` true
Klemme-Klemme-Reaktionszeit			
✔	Knoten	Σ (Gerät-Durchlaufzeit + Steuerungszykluszeit + 2-mal Feldbuszykluszeit + Gerät-Durchlaufzeit)[1]	`IF` Summe < Anforderung `THEN` true `ELSE` false

[1] [vgl. Höme 2013, S. 3]

Vorteile: Dieses Entwurfsmuster besitzt unterschiedliche Vorteile, die im Folgenden aufgelistet werden.

- Ressourcennutzung: Findet die Steuerung aller Mischeinrichtungen auf nur einem Knoten statt, müssen für das spätere Hinzufügen neuer Mischeinrichtungen keine neuen Knoten hinzugefügt werden und an dem bestehenden nur minimale Änderungen vorgenommen werden. Es können leichter neue zu steuernde Sortiereinrichtungen angeschlossen werden, ohne neue Knoten hinzufügen zu müssen.

Nachteile: Die Nachteile des Entwurfsmusters werden im Folgenden aufgelistet.

- Verfügbarkeit: Fällt der Knoten, der für das Mischen von Komponenten zuständig ist, aus, kann trotzdem noch auf die archivierte Anzahl von Komponenten zugegriffen werden.
- Ressourcennutzung: Wird eine neue Sortiereinrichtung hinzugefügt, muss der Knoten angepasst werden.

Anhang E Modelle der fördertechnischen Anlage

Im Nachfolgenden sind die erstellten Modelle für die *Metalltrennungsanlage* (siehe Abschnitt 9.1) dargestellt.

Anhang E 1 Komponentenmodell

Auf Basis der Anlagenbeschreibung und des Anlagenlayouts (siehe Abschnitt 9.1.1) kann die benötigte Automatisierungshardware ermittelt werden. Mithilfe des Anlagenlayouts, das die Hardware grafisch beinhaltet, kann ein Teil der benötigten Hardware ausgewählt werden. Im Anlagenlayout sind beispielsweise die Elemente *Hupe, Detektor, Düse, Motor für die Gurtförderer* sowie *Füllstandsensor* enthalten. Da im Anlagenlayout nicht die gesamte Hardware grafisch dargestellt wird, ist es notwendig, die fehlende Automatisierungshardware anhand der Anlagenbeschreibung, welche die Funktionalität der Anlage beinhaltet, zu ermitteln. In Zusammenhang mit der *Metalltrennungsanlage* sind in der Anlagenbeschreibung deren *Temperatursensoren* enthalten. Das nachfolgende *Komponentenmodell* beinhaltet die für die *Metalltrennungsanlage* relevante Automatisierungshardware.

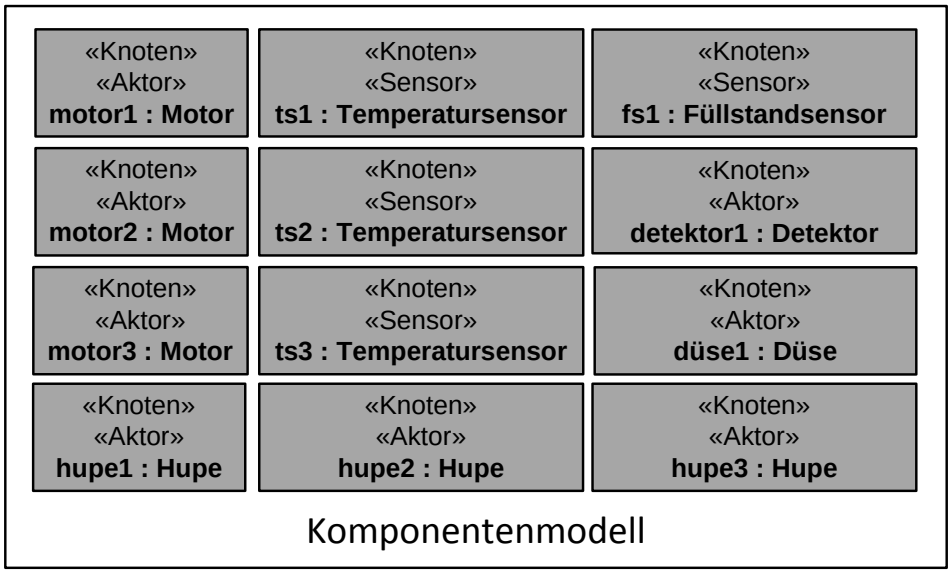

Abbildung E.1: Komponentenmodell der Metalltrennungsanlage

Anhang E 2 Funktionsmodell

Das *Funktionsmodell* kann – wie das *Komponentenmodell* – auf Basis der Anlagenbeschreibung und des Anlagenlayouts (siehe Abschnitt 9.1.1) erstellt werden. Beide beschreiben die Funktionalität – funktionale Anforderungen –, die für die *Metalltrennungsanlage* notwendig ist. Auf Basis dieser Funktionalitäten – *Transportieren, Sortieren, Hupen, Temperatur überwachen, Not-Halt* und *Füllstand überwachen* – können die jeweiligen Funktionsmuster (siehe Anhang C) ausgewählt werden. Das Ergebnis der Anwendung dieser Funktionsmuster ist im *Funktionsmodell* dargestellt. Das nachfolgende *Funktionsmodell* (siehe Abbildung E.2 auf Seite 169) beinhaltet die für die *Metalltrennungsanlage* relevante Funktionalität.

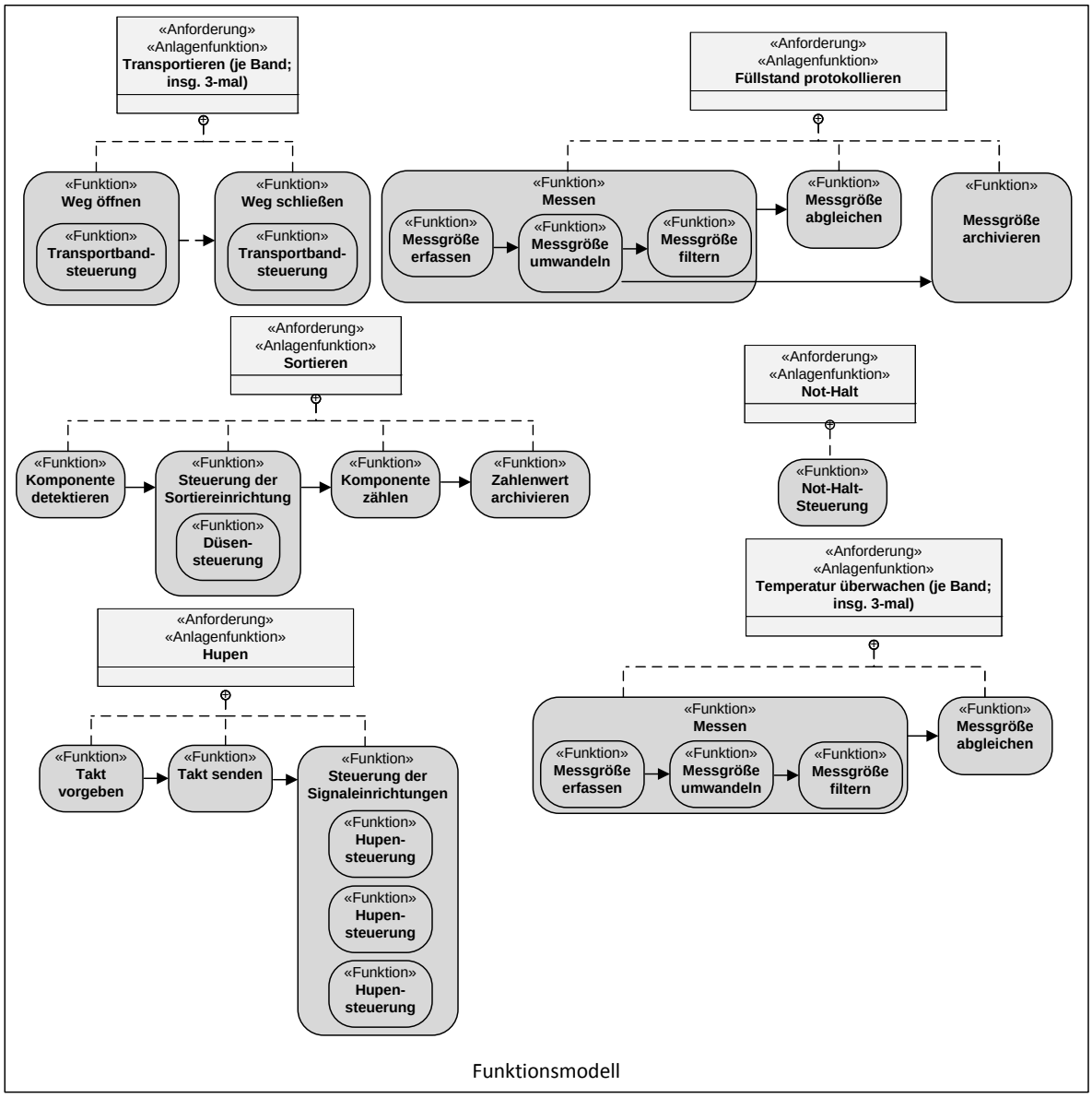

Abbildung E.2: Funktionsmodell der Metalltrennungsanlage

Anhang E 3 Software- und Topologiemodell

Basierend auf der Anlagenbeschreibung, dem Anlagenlayout, dem *Komponentenmodell* sowie dem *Funktionsmodell*, wird die Automatisierungshardware mit der Automatisierungssoftware – Automatisierungsfunktionen in Form von Funktionsblöcken – verknüpft. Hierzu werden Daten- oder Steuerverbindungen verwendet. Im Kontext der *Metalltrennungsanlage* wird die Funktionalität *Transportieren* mit der Automatisierungshardware *Motor* verbunden. Des Weiteren wird die Automatisierungshardware *Düse* sowie *Detektor* mit der Funktionalität *Sortieren* verknüpft. Die Funktionalität *Hupen* wird mit der Automatisierungshardware *Hupe* und die Funktionalität *Temperatur überwachen* wird mit der Automatisierungshardware *Temperatursensor* verbun-

den. Außerdem wird die Automatisierungshardware *Füllstandsensor* mit der Funktionalität *Füllstand überwachen* verknüpft. Zuletzt muss beachtet werden, dass eine Überhitzung der Motoren – Funktionalität *Temperatur überwachen* – sowie eine Übersteigung der Kapazität des Containers – Funktionalität *Füllstand überwachen* – zum Not-Halt führen. Deshalb werden diese Funktionalitäten mit der Funktionalität *Not-Halt* verbunden. In Zusammenhang mit der *Metalltrennungsanlage* wird das Ergebnis dieser Verbindungen zwischen Automatisierungssoftware und -hardware im Folgenden dargestellt (siehe Abbildung E.3 auf Seite 171).

Anhang E 4 Deploymentmodell

Die Anwendung des Verteilungsmusters »getrennte Knoten« für die drei Funktionsbausteine »Transportbandsteuerungen« ist vorteilhaft gegenüber dem Verteilungsmuster »gemeinsamer Knoten für alle Transportbandsteuerungen«, weil die nfA *Zeitverhalten* besser erfüllt wird. Die drei Funktionsbausteine werden jeweils auf die Steuerungen verteilt, die dem Transportweg am nächsten liegen. Dadurch verringert sich der Kommunikationsweg und das *Zeitverhalten* wird optimiert. Eine Verteilung der »Not Halt«- und »Hupensteuerung« auf vier »getrennte Knoten« ist vorteilhaft, weil dadurch ein kurzer Kommunikationsweg bei einer Anlagenstörung vorgegeben wird. Im Notfall werden zur Ansteuerung der Hupe kurze Kommunikationswege über ein Netzwerk benötigt. Dies ist möglich, da die »Hupensteuerungen« jeweils auf die Steuerungen verteilt werden, die der Hupe am nächsten liegen. Des Weiteren ruft die »Not Halt Steuerung« direkt die »Hupensteuerungen« auf.

Die gesamten Sortierfunktionen *(Düsensteuerung, Komponente zählen, Zahlenwert archivieren)* werden zentral auf einem Knoten implementiert, der ein kurzes *Zeitverhalten* ermöglicht und die Speicherkapazität für den zu archivierenden Messwert bereitstellt. Die Anwendung des Verteilungsmusters »getrennte Knoten« für die Funktionsbausteine der Temperaturüberwachung ist vorteilhaft gegenüber dem Verteilungsmuster »gemeinsamer Knoten für alle Temperaturüberwachungen«, weil zur Temperaturüberwachung keine langen Kommunikationswege notwendig sind. Die Funktionsbausteine der Temperaturüberwachung werden jeweils auf die Steuerungen verteilt, die dem Transportweg und somit dem Temperatursensor am nächsten liegen. Die Archivierung des Füllstands im Container findet direkt am Füllstandsensor statt, da dieser Speicherkapazität bereitstellt und keine unnötige Kommunikation stattfindet. Im Nachfolgenden ist das *Deploymentmodell* der *Metalltrennungsanlage* dargestellt (siehe Abbildung E.4 auf Seite 172).

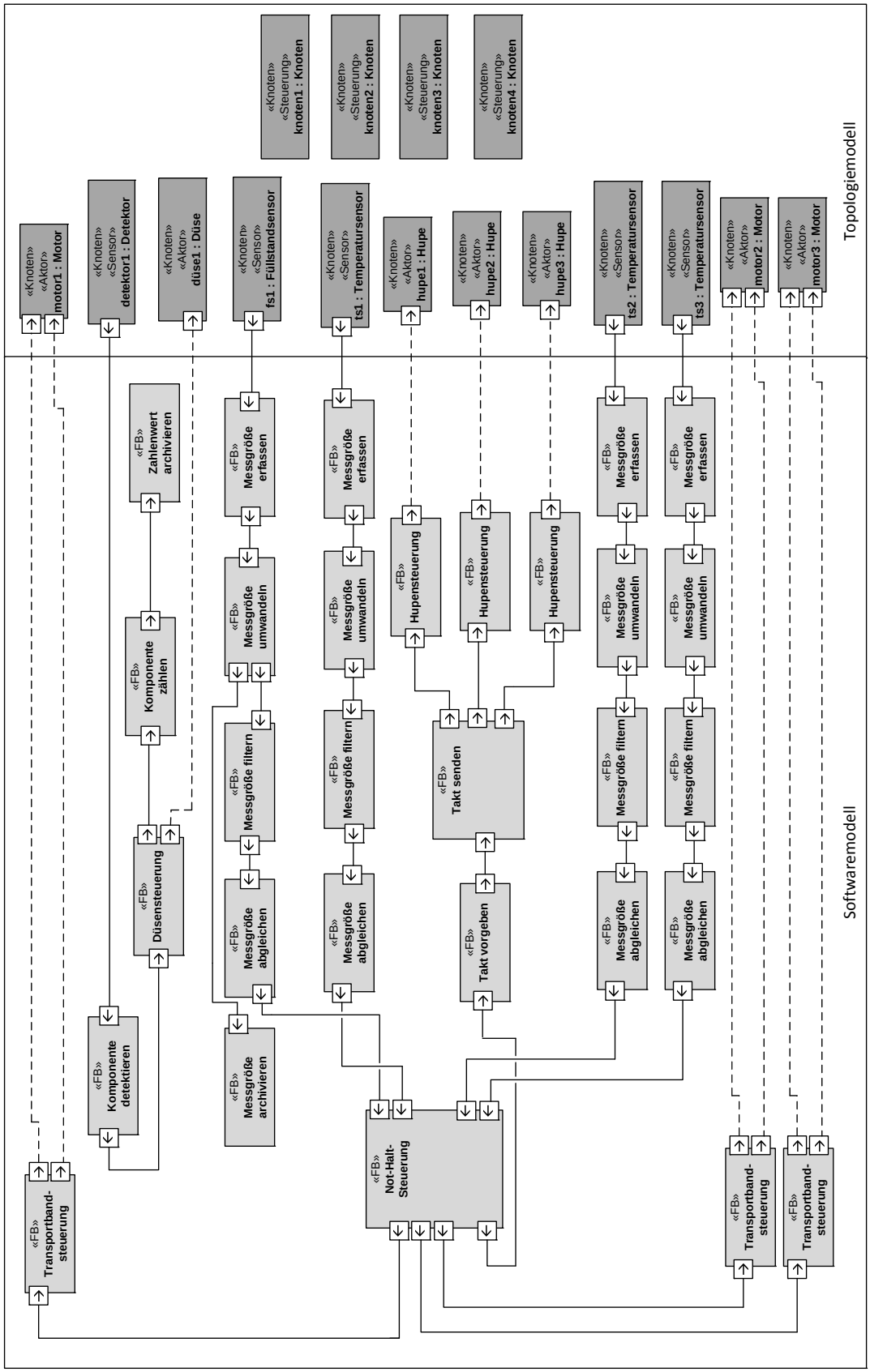

Abbildung E.3: Software- und Topologiemodell der Metalltrennungsanlage

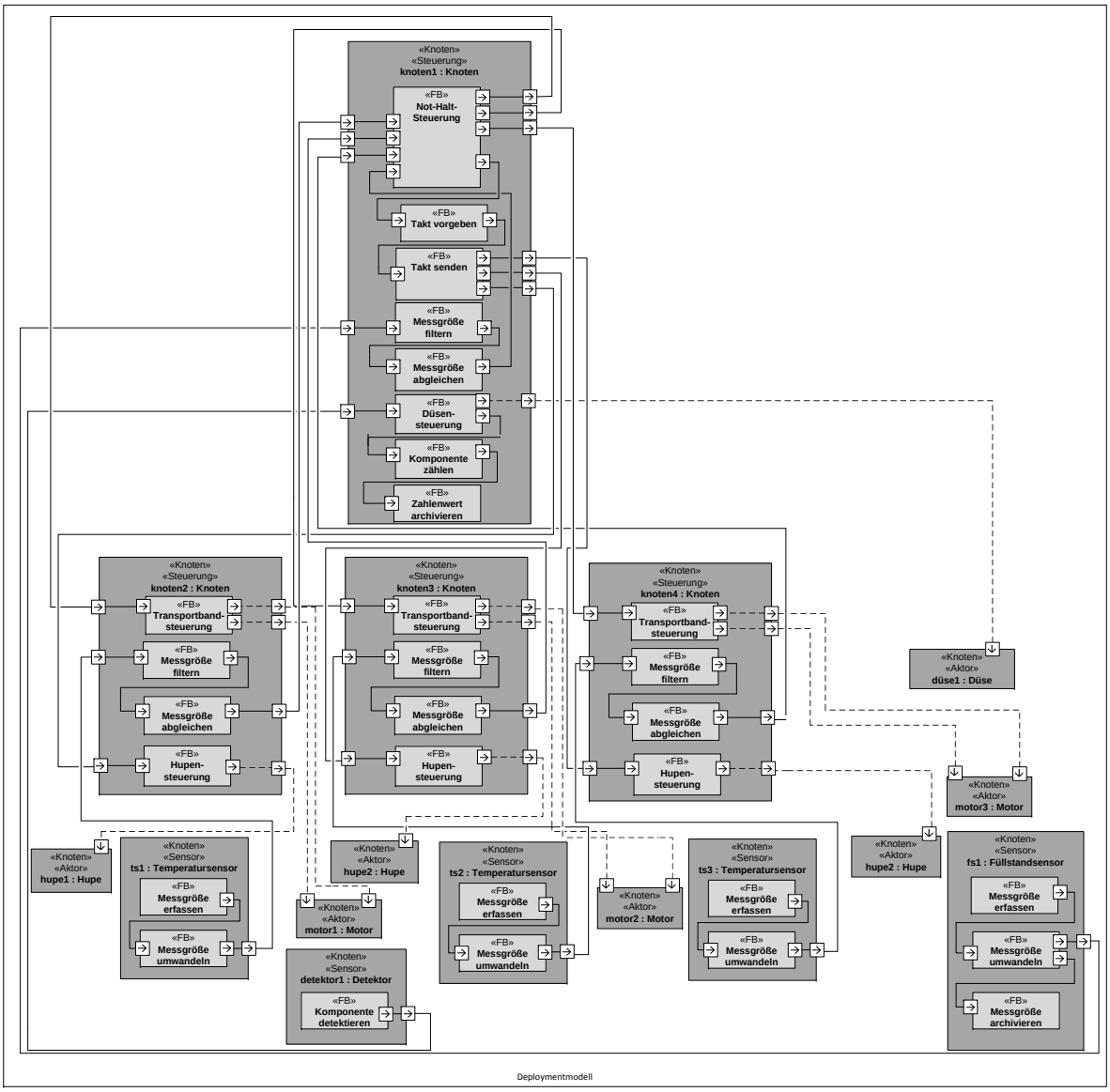

Abbildung E.4: Deploymentmodell der Metalltrennungsanlage

Anhang F Modelle der verfahrenstechnischen Anlage

Im Nachfolgenden sind die erstellten Modelle für die *Mischanlage* (siehe Abschnitt 9.2) dargestellt.

Anhang F 1 Komponentenmodell

Auf Basis des R&I-Fließbilds (siehe Abschnitt 9.2.1) kann die benötigte Automatisierungshardware ermittelt werden. Mithilfe des R&I-Fließbilds, das die Hardware grafisch beinhaltet, kann die benötigte Automatisierungshardware ausgewählt werden. Das R&I-Fließbild beinhaltet alle Elemente, wie beispielsweise *Pumpe, Ventil, Rührer* und *Füllstandsensor*. Im Kontext der *Mischanlage* beinhaltet das nachfolgende *Komponentenmodell* die relevante Automatisierungshardware.

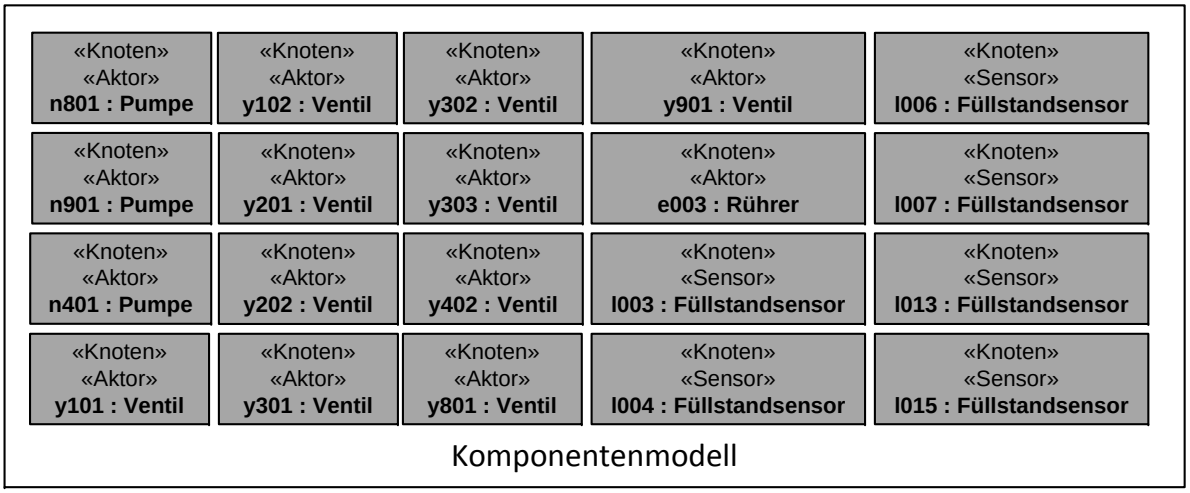

Abbildung F.1: Komponentenmodell der Mischanlage

Anhang F 2 Funktionsmodell

Das *Funktionsmodell* kann – wie das *Komponentenmodell* – auf Basis des R&I-Fließbilds erstellt werden. Zusätzlich muss die Anlagenbeschreibung der *Mischanlage* berücksichtigt werden. Das R&I-Fließbild sowie die Anlagenbeschreibung erläutern die Funktionalität – funktionale Anforderungen –, die für die *Mischanlage* notwendig ist. Auf Basis dieser Funktionalitäten – *Transportieren mit oder ohne Pumpe, Mischen* und *Füllstand überwachen* – können die jeweiligen Funktionsmuster (siehe Anhang D) ausgewählt werden. Das Ergebnis der Anwendung dieser Funktionsmuster ist im *Funktionsmodell* dargestellt. Das nachfolgende *Funktionsmodell* (siehe Abbildung F.2 auf Seite 174) beinhaltet die für die *Mischanlage* relevante Funktionalität.

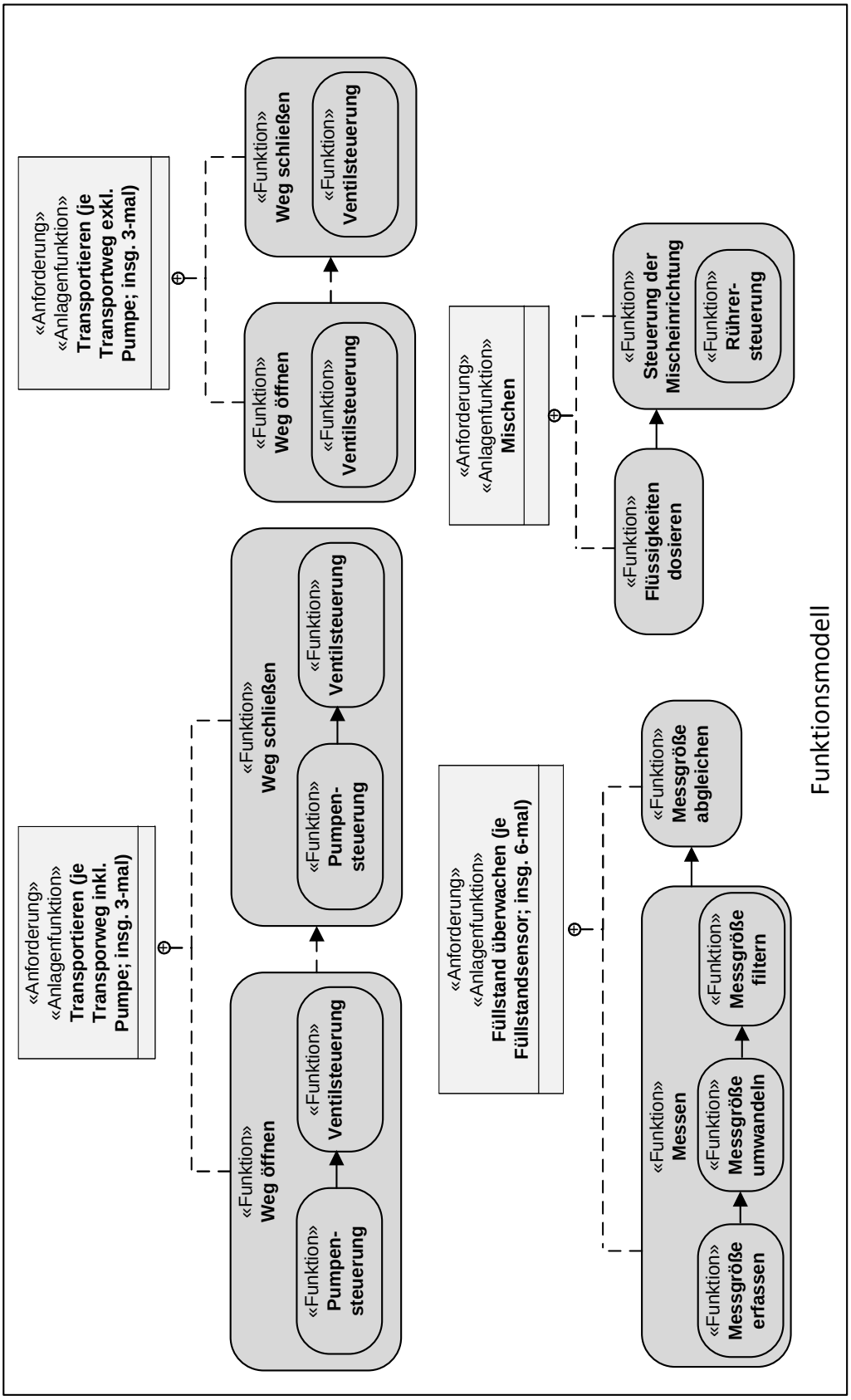

Abbildung F.2: Funktionsmodell der Mischanlage

Anhang F 3 Software- und Topologiemodell

Basierend auf der Anlagenbeschreibung, des R&I-Fließbilds, des *Komponentenmodells* sowie des *Funktionsmodells* wird die Automatisierungshardware mit der Automatisierungssoftware – Automatisierungsfunktionen in Form von Funktionsblöcken – verknüpft. Dazu werden Daten- oder Steuerverbindungen verwendet. Im Kontext der *Mischanlage* wird die Funktionalität des *Transportierens* mit der Automatisierungshardware *Ventil* und bei Bedarf mit der *Pumpe* verbunden. Des Weiteren wird die Automatisierungshardware *Rührer* mit der Funktionalität *Mischen* verknüpft. Die Funktionalität *Füllstand überwachen* wird mit der Automatisierungshardware *Füllstandsensor* verbunden. Zuletzt muss, wenn das Mischverhältnis erreicht wird, beachtet werden, dass der relevante Transportweg geschlossen wird. Deshalb muss die Funktionalität *Mischen* mit der Funktionalität *Transportieren* verbunden werden. Zusätzlich wird die Funktionalität *Transportieren* mit der Funktionalität *Füllstand überwachen* verknüpft, weil dadurch festgelegt wird, wann eine Füllstandüberwachung startet und beim Erreichen des Sollwerts der Transportweg geschlossen wird. In Zusammenhang mit der *Mischanlage* wird das Ergebnis dieser Verbindungen zwischen Automatisierungssoftware und -hardware im folgenden *Topologie- und Softwaremodell Deploymentmodell* der *Metalltrennungsanlage* dargestellt (siehe Abbildung F.3 auf Seite 176) dargestellt.

Anhang F 4 Deploymentmodell

Die Anwendung des Verteilungsmusters »getrennte Knoten« für die Funktionsbausteine »Ventilsteuerung« und »Pumpensteuerung« ist vorteilhaft gegenüber dem Verteilungsmuster »gemeinsamer Knoten für alle Ventil- und Pumpensteuerungen«, weil die nfA *Zeitverhalten* besser erfüllt wird. Die Funktionsbausteine werden jeweils auf die Steuerungen verteilt, die dem Transportweg am nächsten liegen. Dadurch verringert sich der Kommunikationsweg und das *Zeitverhalten* wird optimiert. Eine Verteilung der »Messgröße filtern« und »Messgröße abgleichen« auf die sechs »getrennte Knoten« ist vorteilhaft, weil dadurch ein kurzer Kommunikationsweg zu den jeweiligen »Pumpen- und Ventilsteuerungen« vorgegeben wird. Die gesamte Mischfunktion – Flüssigkeiten dosieren und Rührersteuerung – wird zentral auf einem Knoten implementiert, und zwar auf dem Knoten, welcher der Mischeinrichtung am nächsten liegt. Die »Erfassung und Umwandlung der Messgröße« findet auf den jeweiligen Sensoren statt. In der nachfolgenden Abbildung F.4 auf Seite 177 ist das *Deploymentmodell* der *Mischanlage* dargestellt.

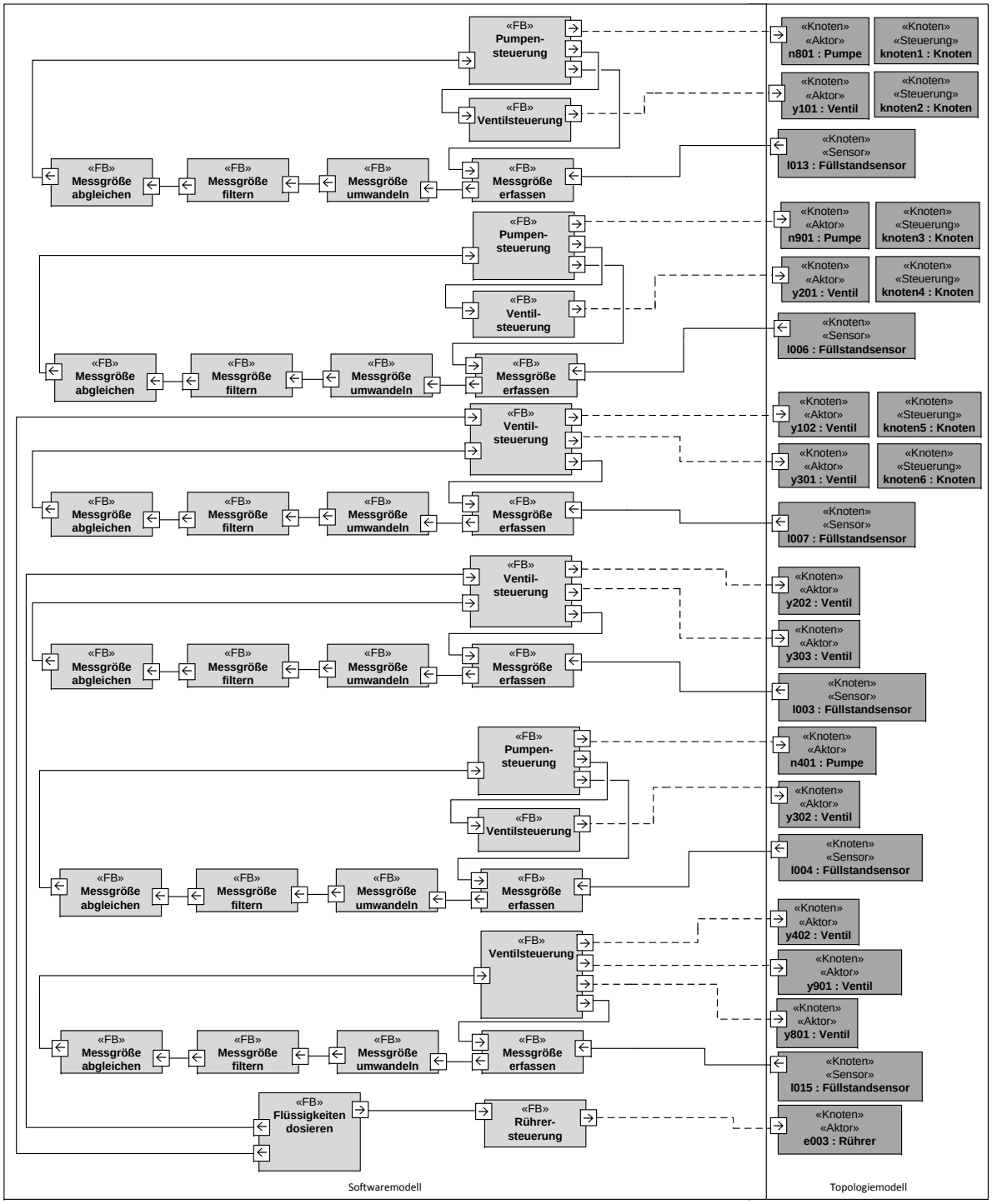

Abbildung F.3: Software- und Topologiemodell der Mischanlage

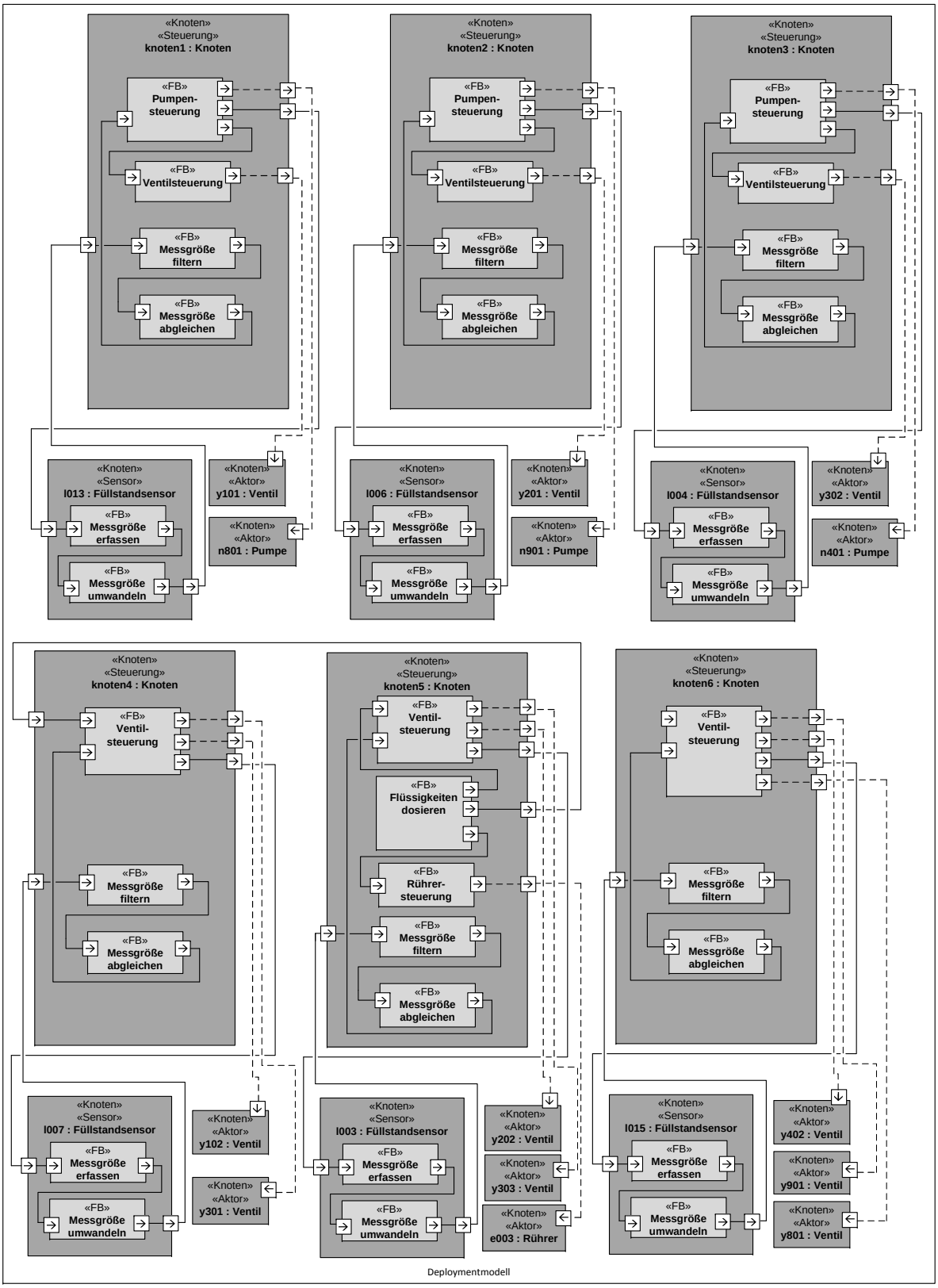

Abbildung F.4: Deploymentmodell der Mischanlage

Glossar

4 + 1-Sichtenkonzept: Das »4 + 1«-Sichtenkonzept [Kruchten 1995], entwickelt von P. B. Kruchten, dient der Beschreibung und Entwicklung von Architekturen softwareintensiver Systeme. Es basiert auf mehreren Sichten, die eine separate Betrachtung von Anliegen ermöglichen. Hierzu werden für jede Sicht unabhängige Architekturbeschreibungen erstellt und die hierfür verwendeten Elemente festgelegt. Das »4 + 1«-Sichtenkonzept besteht aus fünf zentralen Sichten – logische Sicht, Entwicklungssicht, physische Sicht, Prozesssicht und Szenario.

4-Schichtenarchitektur: Die 4-Schichtenarchitektur ist ein Prinzip zur Strukturierung von Softwaresystemen und kommt aus dem Bereich der UML [vgl. Rupp et al. 2007, S. 49 ff.]. Sie besteht aus vier Schichten – M0 Laufzeitschicht, M1 Modellschicht, M2 Metamodellschicht und Meta-Metamodellschicht. Grundsätzlich beschreibt die oberste Schicht die Architektur der darunter liegenden Schicht [vgl. Rupp et al. 2007, S. 49 ff.].

Eingebettete Systeme: Ein eingebettetes System ist ein informationsverarbeitendes System, das in einen technischen Kontext eingebunden wird [vgl. Marwedel 2008, S. 1]. Zumeist sind eingebettete Systeme mit einer physikalischen Umwelt verbunden (über Sensoren und Aktoren) [vgl. Marwedel 2008, S. 2] und übernehmen im technischen Kontext oftmals Steuerungs-, Regelungs-, Datenverarbeitungs- und Überwachungsaufgaben [vgl. Scholz 2005, S. 1].

Feldgerät: Unter dem Begriff Feldgerät werden grundsätzlich die Sensoren und Aktoren eines Automatisierungssystems zusammengefasst [vgl. Birkhofer 2001, S. 19]. Die Sensoren sammeln Informationen über die Umwelt und die Aktoren beeinflussen die Umwelt [vgl. Marwedel 2008, S. 2]. Die Hauptaufgabe von Feldgeräten ist es, die steuernden Komponenten mit Prozesswerten zu versorgen beziehungsweise Steuerbefehle dieser Einheiten durch Stelleingriffe in den Prozess umzusetzen [vgl. Birkhofer 2001, S. 19].

Generisches Vorgehensmodell: Ein generisches Vorgehensmodell kann auf die projektspezifischen Gegebenheiten adaptiert werden und ist somit auf verschiedene Situationen beziehungsweise Problemstellungen anwendbar.

Gerichtete Graphen: Ein gerichteter Graph stellt eine Menge an Objekten sowie die dazwischen bestehende Verbindung dar. Diese Objekte werden Knoten und die Verbindungen zwischen diesen Knoten Kanten genannt [vgl. Fischbach 2008, S. 65 ff.]. Die Kanten zwischen den Knoten können gerichtet oder ungerichtet modelliert werden, wobei dies von der Art der Verbindung abhängt [vgl. Fischbach 2008, S. 43]. Die Kanten eines gerichteten Graphen werden als Pfeile dargestellt und zeigen von einem ersten zu einem zweiten Knoten [vgl. Fischbach 2008, S. 44].

Intelligentes Feldgerät: Unter dem Begriff intelligentes Feldgerät werden grundsätzlich Sensoren und Aktoren zusammengefasst, die über einen Mikroprozessor verfügen und somit Rechenressourcen zur Vorverarbeitung der Signale im Feldgerät besitzen [vgl. Felleisen 2001, S. 273].

Modell: Die VDI 3633-Richtlinie beschreibt ein Modell als eine vereinfachte Nachbildung eines geplanten oder existierenden Systems [vgl. VDI 3633 2000, S. 23][#]. Hierbei werden die benötigten Prozesse in einem anderen begrifflichen oder gegenständlichen System nachgebildet [vgl. VDI 3633 2000, S. 23][#].

Monokriterielles Optimierungsproblem: Ein monokriterielles Optimierungsproblem ist dadurch charakterisiert, dass von den erwähnten Zielkriterien lediglich eins bei der Problemlösung Berücksichtigung findet [vgl. Wenger 2010, S. 65 f.]. Hierbei wird der Wert der besten Lösung gesucht.

Pareto-optimale-Lösung: Die Pareto-optimale-Lösung ist eine Lösung aus einer gegebenen Lösungsmenge, zu der es keine Lösung gibt, die in mindestens einer Zielgröße besser ist und in keiner Zielgröße schlechter [vgl. Biesenbach 2007, S. 176]. Dies bedeutet, dass sich innerhalb einer Lösungsmenge keine Zielgröße verbessern lässt, ohne mindestens eine andere Zielgröße zu verschlechtern [vgl. Biesenbach 2007, S. 176].

Sicht: Eine Sicht ist die Darstellung der Architektur eines Systems aus der Perspektive einer Menge von Sachverhalten und beinhaltet zumeist spezifische Informationen [vgl. ISO/IEC 42 010 2011, S. 3][#].

Quellenverzeichnis

- Nicht von der Verfasserin stammende Literatur, die mit [<Kurzbeleg>] zitiert wurde, ist in der Liste »Literaturverzeichnis« aufgeführt.
- Zitierte Normen, Richtlinien und Empfehlungen finden sich in der Liste »Referenzierte Normen, Richtlinien und Empfehlungen« (siehe Seite 192) und sind durch die Beschreibung [<Typ><Nummer>]# erkennbar.
- Die verwendeten Internetquellen befinden sich in der Liste »Referenzierte Internetquellen« (siehe Seite 193) und sind als [<Kurzbeleg>]@ zitiert.
- Die mit [<Kurzbeleg>]* zitierte Literatur der Verfasserin findet sich in der Liste »Veröffentlichungen der Verfasserin« (siehe Seite 194 f.).

Literaturverzeichnis

[Alexander 1964] C. Alexander. *Notes on the Syntheses of Form.* 7. Aufl. Cambridge Massachusetts London: Harvard University Press, 1964.

[Alexander et al. 1977] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobsen, I. Fiksdahl-King und S. Angel. *A Pattern Language: Towns – Buildings – Construction.* New York: Oxford University Press, 1977.

[Altiparmak & Dengiz 2009] F. Altiparmak und B. Dengiz. »A cross entropy approach to design of reliable networks«. In: *European Journal of Operational Research.* Jahrgang 199, Heft 2 (2009), S. 542–552.

[Altiparmak et al. 2009] F. Altiparmak, B. Dengiz und A. E. Smith. »A General Neural Network Model for Estimating Telecommunications Network Reliability«. In: *IEEE Transactions on Reliability.* Jahrgang 58, Heft 1 (2009), S. 2–9.

[Alznauer et al. 2003] R. Alznauer, K. Auer und A. Fay. »Wiederverwendung von Automatisierungs-Informationen und -Lösungen«. In: *atp-edition – Automatisierungstechnische Praxis.* Heft 3 (2003), S. 31–35.

[Armoush et al. 2009] A. Armoush, F. Salewski und S. Kowalewski. »Design Pattern Representation for Safety-Critical Embedded Systems«. In: *Software Engineering & Applications.* Heft 2 (2009), S. 1–12.

[Bangemann et al. 2008] T. Bangemann, C. Diedrich, A. W. Colombo und S. Karnouskos. »SOCRADES – Service Oriented Architecture in der Automatisierungstechnik«. In: *Tagungsband »Automation 2008«.* Baden-Baden, Germany, 3. – 4. Juni 2008.

[Bangemann et al. 2012] T. Bangemann, N. Suchold, A. W. Colombo und S. Karnouskos. »Die Integration Service orientierter Architekturen in der Automation«. In: *Tagungsband »Automation 2012«.* Baden-Baden, Germany, 13. – 14. Juni 2012.

[Barth 2011] M. Barth. »Automatisch generierte Simulationsmodelle verfahrenstechnischer Anlagen für den Steuerungstest«. VDI, Düsseldorf, Fortschrittsberichte, Reihe 20, Rechnerunterstützte Verfahren, Nr. 438. Dissertation. Hamburg: Fakultät für Maschinenbau, Helmut-Schmidt-Universität, 2011.

[Basile et al. 2011] F. Basile, P. Chiacchio und D. Gerbasio. »Progress in PLC programming for distributed automation systems control«. In: *Tagungsband der 9. Tagung »IEEE International Conference on Industrial Informatics« (INDIN).* Lisbon, Portugal, 26. – 29. Juli 2011, S. 621–627.

[Becker & Pereira 2002] L. B. Becker und C. E. Pereira. »SIMOO-RT-an object-oriented framework for the development of real-time industrial automation systems«. In: *IEEE Transactions on Robotics and Automation.* Jahrgang 18, Heft 4 (2002), S. 421–430.

[Beder et al. 2000] D. M. Beder, A. Romanovsky, B. Randell, C. R. Snow und R. J. Stroud. »An application of fault tolerance patterns and coordinated atomic actions to a problem in railway scheduling«. In: *ACM Operating System Review.* Jahrgang 34, Heft 4 (2000), S. 21–31.

[Bender 2005] K. Bender. *Embedded Systems – qualitätsorientierte Entwicklung.* Berlin Heidelberg: Springer, 2005.

[Biesenbach 2007] A. Biesenbach. *Multi-Site-Scheduling in der chemischen Industrie: Anlagenbelegungsplanung bei international verteilten Produktionsstandorten.* Wiesbaden: GWV Fachverlag, 2007.

[Birkhofer 2001] R. Birkhofer. »Modellbasierte Beschreibung zur offenen Integration intelligenter Feldgeräte der Automatisierungstechnik«. K. Bender (Hrsg.), Herbert Utz, München. Dissertation. München: Informationstechnik im Maschinenwesen, 2001.

[Brandl 2006] D. Brandl. *Design patterns for flexible manufacturing.* USA: ISA, 2006.

[Broy 2010] M. Broy. »Cyber-Physical Systems – Wissenschaftliche Herausforderungen bei der Entwicklung«. In: *Cyber-Physical Systems.* Hrsg. von M. Broy. acatech DISKUTIERT. Berlin Heidelberg: Springer, 2010, S. 17–31.

[Broy et al. 1999] M. Broy, F. Huber und B. Schätz. »AutoFocus – Ein Werkzeugprototyp zur Entwicklung eingebetteter Systeme«. In: *Informatik Forschung und Entwicklung.* Jahrgang 14, Heft 3 (1999), S. 121–134.

[Catalan et al. 2011] C. Catalan, F. Serna, A. Blesa, J. M. Colom und J. M. Rams. »COSME: A distributed control platform for communicating machine tools in Agile Manufacturing Systems«. In: *Tagungsband der 16. Tagung »IEEE International Conference on Emerging Technologies and Factory Automation« (ETFA).* Toulouse, France, 5. – 9. September 2011, S. 1–8.

[Cengic et al. 2006] G. Cengic, O. Ljungkrantz und K. Akesson. »A Framework for Component Based Distributed Control Software Development Using IEC 61 499«. In: *Tagungsband der 11. Tagung »IEEE International Conference on Emerging Technologies and Factory Automation« (ETFA).* Prague, Czech Republic, 20. – 22. September 2006, S. 782–789.

[Christensen 2000] J. H. Christensen. »Design patterns for system engineering in IEC 61 499«. In: *Fachtagung »Verteilte Automatisierung«.* Magdeburg, Germany, 22. – 23. März 2000, S. 63–71.

[Davis & Leffingwell 1996] A. M. Davis und D. A. Leffingwell. »Using Requirements Management to Speed Delivery of Higher Quality Applications«. In: *Rational Software.* (1996), S. 1–14.

[Dechema 2009] Gesellschaft für Chemische Technik und Biotechnologie e. v. »Trendbericht: Chemieanlagen-Konzepte – Welche Entwicklungen bestimmen den Chemieanlagenbau?« In: *Achema magazine.* (2009), S. 34–38.

[Diedrich et al. 2011] C. Diedrich, A. Lüder und L. Hundt. »Bedeutung der Interoperabilität bei Entwurf und Nutzung von automatisierten Produktionssystemen«. In: *at – Automatisierungstechnik* Jahrgang 59, Heft 7 (2011), S. 426–438.

[Diekhoff 2010] D. Diekhoff. »AUTOSAR basic software for complex control units«. In: *Tagungsband der Tagung »IEEE International Conference on Design, Automation & Test in Europe« (DATE).* Dresden, Germany, 8. – 12. März 2010, S. 263–266.

[Djambova 2005] T. Djambova. »Ein Beitrag zur objektorientierten Modellierung von Automatisierungssystemen mit UML«. ISLE, Illmenau. Dissertation. Illmenau: Institut für Informatik und Softwaretechnik, Technische Universität Illmenau, 2005.

[Douglass 2003] B. P. Douglass. *Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems.* Boston: Addison-Wesley, 2003.

[Dujmović 2002] S. Dujmović. »Anwendungsentwicklung mit Komponenten-Frameworks in der Automatisierungstechnik«. IAS-Forschungsberichte, P. Göhner (Hrsg.), Shaker, Aachen, Band 1/2002. Dissertation. Stuttgart: Institut für Automatisierungs- und Softwaretechnik, Universität Stuttgart, 2002.

[Dutta & Mitra 1993] A. Dutta und S. Mitra. »Integrating heuristic knowledge and optimization models for communication network design«. In: *IEEE Transactions on Knowledge and Data Engineering.* Jahrgang 5, Heft 6 (1993), S. 999–1017.

[Eberle & Göhner 2003] S. Eberle und P. Göhner. »Softwareentwicklung für eingebettete Systeme mit strukturierten Komponenten – Teil 2: Komponentenorientierte Modellierung und Realisierung«. In: *atp-edition – Automatisierungstechnische Praxis.* Jahrgang 45, Heft 2 (2003), S. 61–73.

[Eberle & Göhner 2004] S. Eberle und P. Göhner. »Softwareentwicklung für eingebettete Systeme mit strukturierten Komponenten – Teil 1: Komponentenorientierte Zerlegung«. In: *atp-edition – Automatisierungstechnische Praxis.* Jahrgang 46, Heft 3 (2004), S. 41–52.

[Eisemann et al. 2009] U. Eisemann, D. Stichling und J. Stroop. »Effiziente Software-Entwicklung und -Verifikation in einer Autosar-Werkzeugkette«. In: *ATZelektronik.* Jahrgang 4, Heft 3 (2009), S. 50–55.

[EIT Bericht 2010] *Forschungsbericht 2010.* Magdeburg: Fakultät für Elektrotechnik und Informationstechnik, Otto-von-Guericke-Universität, 2010.

[equal] K. Bender, S. Dominka, A. Koc, M. Pöschl, M. Russ und B. Stützel. *Embedded Systems – qualitätsorientierte Entwicklung.* Ergebnisse des Projekts EQUAL. Berlin Heidelberg: Springer, 2005.

[Fantuzzi et al. 2009] C. Fantuzzi, C. Bonfe und C. Secchi. »A design pattern for model based software development for automatic machinery«. In: *Tagungsband der 13. Tagung »IFAC Symposium on Information Control Problems in Manufacturing« (INCOM).* Moskau, Russland, 3. – 5. Juni 2009.

[Favre-Bulle 2004] B. Favre-Bulle. *Automatisierung komplexer Industrieprozesse: Systeme, Verfahren und Informationsmanagement.* Wien: Springer, 2004.

[Fay 2005] A. Fay. »Engineering in vernetzten, offenen, durchgängigen Systemen«. In: *at – Automatisierungstechnik* Jahrgang 53, Heft 4–5 (2005), S. 205–210.

[Fay 2009] A. Fay. »Effizientes Engineering komplexer Automatisierungssysteme«. In: *Wird der Verkehr automatisch sicherer? Beschreibungsmittel, Methoden und Werkzeuge des integrierten Systementwurfs zur Fahrzeug- und Verkehrsautomatisierung.* Hrsg. von E. Schnieder. Braunschweig: Technische Universität Braunschweig, Institut für Verkehrssicherheit und Automatisierungstechnik, 2009, S. 29–37.

[Fay et al. 2009] A. Fay, M. Schleipen und M. Mühlhause. »Wie kann man den Engineering-Prozess systematisch verbessern?« In: *atp-edition – Automatisierungstechnische Praxis.* Heft 1-2 (2009), S. 80–85.

[Feldhorst & Libert 2010] S. Feldhorst und S. Libert. »Software-Methoden für die Automatisierung«. In: *Internet der Dinge in der Intralogistik.* Hrsg. von W. Günthner und M. ten Hompel. VDI-Buch. Berlin Heidelberg: Springer, 2010, S. 29–37.

[Felleisen 2001] M. Felleisen. *Prozessleittechnik für die Verfahrensindustrie.* München: Oldenbourg Industrieverlag, 2001.

[Felser 2003] M. Felser. »Distributed Automation Systems: Herstellerübergreifende Programmierung von verteilten Steuerungen«. In: *BULLETIN SEVelectrosuisse.* Heft 17 (2003), S. 11–15.

[Fencl et al. 2011] T. Fencl, P. Burget und J. Bilek. »Network topology design«. In: *Control Engineering Practice.* Jahrgang 19, Heft 11 (2011), S. 1287–1296.

[Fischbach 2008] K. Fischbach. *Strukturbildung in Peer-to-Peer-Netzwerken.* Köln: Kölner Wissenschaftsverlag, 2008.

[Fletcher & Cleland 2006] J. Fletcher und J. Cleland-Huang. »Softgoal Traceability Patterns«. In: *Tagungsband der 17. Tagung »IEEE International Symposium on Software Reliability Engineering« (ISSRE).* Raleigh, North Carolina, USA, 7. – 10. November 2006, S. 363–374.

[Folmer et al. 2012] J. Folmer, D. Schütz, M. Schraufstetter und B. Vogel-Heuser. »Konzept zur Erhöhung der Flexibilität von Produktionsanlagen durch Einsatz von rekonfigurierbaren Anlagenkomponenten und echtzeitfähigen Softwareagenten«. In: *Herausforderungen durch Echtzeitbetrieb.* Hrsg. von W. A. Halang. Informatik aktuell. Berlin Heidelberg: Springer, 2012, S. 121–130.

[Frank et al. 2013b] T. Frank, D. Schütz und B. Vogel-Heuser. »Funktionaler Anwendungsentwurf für agentenbasierte, verteilte Automatisierungssysteme«. In: *Agentensysteme in der Automatisierungstechnik.* Hrsg. von P. Göhner. Xpert.press. Berlin Heidelberg: Springer, 2013, S. 3–19.

[Frank 2014] T. Frank. »Entwicklung und Evaluation einer Modellierungssprache für den Architekturentwurf von verteilten Automatisierungsanlagen auf Basis der Systems Modeling Language (SysML)«. Dissertation. München: Lehrstuhl für Automatisierung und Informationssysteme, Technische Universität München, 2104.

[Frey 2004] G. Frey. »Spezifikation von Betriebszuständen und Betriebsartenumschaltungen – GEMMA: Ein Ansatz aus Frankreich«. In: *atp-edition – Automatisierungstechnische Praxis.* Jahrgang 46, Heft 3 (2004), S. 37–40.

[Frey & Thramboulidis 2011] G. Frey und K. Thramboulidis. »Einbindung der IEC 61 131 in modellgetriebene Entwicklungsprozesse«. In: *Tagungsband »Automation 2011«.* Baden-Baden, Germany, 28. – 29. Juni 2011.

[Friedrich et al. 2009] J. Friedrich, U. Hammerschall, M. Kuhrmann und M. Sihling. *Das V-Modell XT.* 2. Aufl. Berlin Heidelberg: Springer, 2009.

[Fuchs et al. 2012] J. Fuchs, S. Feldmann und B. Vogel-Heuser. »Modularität im Maschinen- und Anlagenbau – Analyse der Anforderungen und Herausforderungen im industriellen Einsatz«. In: *Tagungsband »Entwurf komplexer Automatisierungssysteme« (EKA).* Magdeburg, Germany, 8. – 10. Mai 2012.

[Fuchs & Vogel-Heuser 2012] J. Fuchs und B. Vogel-Heuser. »Metriken und Methoden zur Umstrukturierung einer modularen Steuerungssoftware im Sondermaschinenbau«. In: *Tagungsband »Automation 2012«.* Baden-Baden, Germany, 13. – 14. Juni 2012.

[Gamma et al. 2004] E. Gamma, R. Helm, R. Johnson und J. Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software.* München: Addison-Wesley, 2004.

[Göhner et al. 2003] P. Göhner, T. Wagner und P. G. de A. Urbano. »Softwareagenten – Einführung und Überblick über eine alternative Art der Softwareentwicklung«. In: *atp-edition – Automatisierungstechnische Praxis (Sonderdruck).* Heft 4 (2003), S. 1–31.

[Göpfert 2009] J. Göpfert. *Modulare Produktentwicklung: Zur gemeinsamen Gestaltung von Technik und Organisation; Theorie, Methodik, Praxis.* Norderstedt: Books on Demand, 2009.

[Grabmair et al. 2002] G. Grabmair, R. Froschauer, T. Strasser und A. Zoitl. »Requirements patterns for embedded systems«. In: *Tagungsband der 10. Tagung »IEEE International Conference on Requirements Engineering« (RE).* Essen, Germany, 9. – 13. September 2002, S. 127–136.

[Gross & Yu 2001] D. Gross und E. Yu. »From non-functional requirements to design through patterns«. In: *Requirements Engineering.* Jahrgang 6, Heft 1 (2001), S. 18–36.

[Hadlich 2015] T. Hadlich. »Verwendung von Merkmalen im Engineering von Systemen«. Dissertation. Magdeburg: Fakultät für Elektrotechnik und Informationstechnik, Otto-von-Guericke-Universität, 2015.

[Harbach et al. 2007] F. Harbach, K. Janschek, U. Jumar, O. Sawodny und G. U. Spohr. »Herausfordernde Anwendungsgebiete der Automatisierungstechnik«. In: *at – Automatisierungstechnik* Jahrgang 55, Heft 5 (2007), S. 260–265.

[Helmus 2003] F. Helmus. *Anlagenplanung: Von der Anfrage bis zur Abnahme.* Weinheim: Wiley-VCH, 2003.

[Herrero & Martinez 2010] D. Herrero-Perez und H. Martinez-Barbera. »Modeling Distributed Transportation Systems Composed of Flexible Automated Guided Vehicles in Flexible Manufacturing Systems«. In: *Tagungsband der 8. Tagung »IEEE International Conference on Industrial Informatics« (INDIN).* Osaka, Japan, 13. – 16. Juli 2010, S. 166–180.

[Holm et al. 2013] T. Holm, S. Schröck, A. Fay, T. Jäger und U. Löwen. »Engineering von "Mechatronik und Software" in automatisierten Anlagen – Anforderungen und Stand der Technik«. In: *Tagungsband »ENVISION2020 – Zukunft der Entwicklung softwareintensiver, eingebetteter Systeme«.* Aachen, Germany, 27. Februar – 1. März 2013.

[Höme 2013] S. Höme, C. Diedrich, M. Damm und T. Werner. »Performancebenchmark für Steuerungen mit synchroner Kommunikation«. In: *Tagungsband »Automation 2013«.* Baden-Baden, Germany, 25. – 26. Juni 2013.

[Hussain & Frey 2008] T. Hussain und G. Frey. »Entwicklung verteilter Steuerungen mit UML und IEC 61499«. In: *Tagungsband »Automation 2008«.* Baden-Baden, Germany, 3. – 4. Juni 2008.

[Ko et al. 1997] K. T. Ko, K. S. Tang, C. Y. Chan, K. F. Man und S. Kwong. »Using genetic algorithms to design mesh network«. In: *Computer.* Jahrgang 30, Heft 8 (1997), S. 56–61.

[Kruchten 1995] P. B. Kruchten. »The 4 + 1 View Model of architecture«. In: *IEEE Software.* Jahrgang 12, Heft 6 (1995), S. 42–50.

[Kuhrmann 2008] M. Kuhrmann. »Konstruktion modularer Vorgehensmodelle: methodisches Erstellen und Pflegen von Entwicklungsstandards und Vorgehensmodellen für Prozessingenieure«. Dissertation. München: Fakultät für Informatik, Technische Universität München, 2008.

[Kumar et al. 1998] G. Kumar, N. Narang und C. P. Ravikumar. »Efficient algorithms for delay-bounded minimum cost path problem in communication networks«. In: *Tagungsband der 5. Tagung »IEEE International Conference on High Performance Computing« (HIPC).* Madras, India, 17. – 20. Dezember 1998, S. 141–146.

[Kurschl 2000] W. Kurschl. »Monitoring von verteilten Systemen«. Dissertation. Linz: Fakultät für Sozial- und Wirtschaftswissenschaften, Johannes Kepler Universtät Linz, 2000.

[Lauber & Göhner 1999] R. Lauber und P. Göhner. *Prozeßautomatisierung 1: Automatisierungssysteme und -strukturen, Computer- und Bussysteme für die Anlagen- und Produktautomatisierung. Echtzeitprogrammierung und Echtzeitbetriebssysteme, Zuverlässigkeits- und Sicherheitstechnik.* 3. Aufl. Berlin Heidelberg: Springer, 1999.

[Lea 1994] D. Lea. *Design Patterns for Avionics Control Systems.* DSSA ADAGE Project, State University of New York, 1994.

[Lin & Yeh 2010] Y. K. Lin und C. T. Yeh. »Evaluation of Optimal Network Reliability Under Components-Assignments Subject to a Transmission Budget«. In: *IEEE Transactions on Reliability.* Jahrgang 59, Heft 3 (2010), S. 539–550.

[Lindemann 2009] U. Lindemann. *Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden.* 3. Aufl. VDI-Buch. Berlin Heidelberg: Springer, 2009.

[Lindner 1996] U. Lindner. »Massive Wiederverwendung: Konzepte, Techniken und Organisation«. In: *OBJEKTspektrum.* Heft 1 (1996), S. 10–19.

[Löwen et al. 2005] U. Löwen, R. Bertsch, B. Böhm, S. Prummer und T. Tetzner. »Systematisierung des Engineerings von Industrieanlagen«. In: *atp-edition – Automatisierungstechnische Praxis.* Jahrgang 47, Heft 4 (2005), S. 54–61.

[Lüder 2006] A. Lüder. »Strukturen zur verteilten Steuerung von Produktionssystemen«. Habilitationsschrift. Magdeburg: Fakultät für Maschinenbau, Otto-von-Guericke-Universität, 2006.

[Lüder et al. 2010] A. Lüder, J. Peschke und R. Sanz. »Design Patterns for Distributed Control Applications«. In: *Distributed Manufacturing.* Hrsg. von H. Kühnle. London: Springer, 2010, S. 155–175.

[Maga & Jazdi 2010] C. Maga und N. Jazdi. »Klassifizierung möglicher Beziehungen zwischen wiederverwendbaren Artefakten in der Automatisierungstechnik«. In: *Tagungsband »Entwurf komplexer Automatisierungssysteme« (EKA).* Magdeburg, Germany, 25. – 27. Mai 2010.

[Maga et al. 2011] C. Maga, N. Jazdi und P. Göhner. »Requirements on Engineering Tools for Increasing Reuse in Industrial Automation«. In: *Tagungsband der 16. Tagung »IEEE, International Conference on Emerging Technologies and Factory Automation« (ETFA).* Toulouse, France, 5. – 9. September 2011, S. 1–7.

[Mandel 2009] S. Mandel. »Proaktive Anlaufabsicherung automatisierter Produktionsanlagen«. VDI, Düsseldorf, Fortschrittsberichte, Reihe 20, Rechnerunterstützte Verfahren, Nr. 426. Dissertation. Hamburg: Fakultät für Maschinenbau, Helmut-Schmidt-Universität, 2009.

[Marquardt & Nagel 2003] W. Marquardt und M. Nagel. »Arbeitsprozessorientierte Unterstützung verfahrenstechnischer Entwicklungsprozesse«. In: *atp-edition – Automatisierungstechnische Praxis.* Heft 4 (2003), S. 52–58.

[Marwedel 2008] P. Marwedel. *Eingebettete Systeme.* Berlin Heidelberg: Springer, 2008.

[Meier 2001] H. Meier. »Verteilte, kooperative Steuerung maschinennaher Abläufe«. Dissertation. München: Fakultät für Maschinenwesen, Technische Universität München, 2001.

[Melzer 2010] I. Melzer. *Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis.* 4. Aufl. Heidelberg: Spektrum, 2010.

[Misra et al. 2009] S. Misra, X. Guoliang und Y. Dejun. »Polynomial Time Approximations for Multi-Path Routing with Bandwidth and Delay Constraints«. In: *Tagungsband der 28. Tagung »IEEE International Conference on Computer Communications« (INFOCOM).* Rio de Janeiro, Brazil, 19. – 25. April 2009, S. 558–566.

[Neumann et al. 2000] P. Neumann, E. E. Grötsch, C. Lubkoll und R. Simon. *SPS-Standard: IEC 61 131 – Programmierung in verteilten Automatisierungssystemen.* 3. Aufl. München: Oldenbourg Industrieverlag, 2000.

[Nüttgens & Rump 2002] M. Nüttgens und F. J. Rump. »Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK)«. In: *Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise).* (2002), S. 64–77.

[Obst et al. 2013] M. Obst, T. Holm, S. Bleuel, U. Claussnitzer, L. Evertz, T. Jäger, T. Nekolla, S. Pech, S. Schmitz und L. Urbas. »Automatisierung im Life Cycle modularer Anlagen«. In: *atp-edition – Automatisierungstechnische Praxis.* Heft 1–2 (2013), S. 24–31.

[Panjaitan & Frey 2006] S. Panjaitan und G. Frey. »Combination of UML Modeling and the IEC 61 499 Function Block Concept for the Development of Distributed Automation Systems«. In: *Tagungsband der 11. Tagung »IEEE International Conference on Emerging Technologies and Factory Automation« (ETFA).* Prague, Czech Republic, 20. – 22. September 2006, S. 766–773.

[Papenfort 2006] J. Papenfort. »Softwarekonzepte für zentrale und dezentrale Steuerungsarchitekturen«. In: *atp-edition – Automatisierungstechnische Praxis.* Jahrgang 48, Heft 1 (2006), S. 42–47.

[Pech & Göhner 2011] S. Pech und P. Göhner. »Flexible dezentrale Automatisierungs- und Regelungssysteme auf Basis von Softwareagenten«. In: *45. Regelungstechnisches Kolloquium* (2011).

[Peng et al. 2010] W. Peng, H. Li, M. Yao und Z. Sun. »Deployment optimization for AUTOSAR system configuration«. In: *Tagungsband der 2. Tagung »IEEE International Conference on Computer Engineering« (ICCET).* Chengdu, China, 16. – 18. April 2010, S. 189–193.

[Prayati et al. 2004] A. Prayati, C. Koulamas, S. Koubias und G. Papadopoulos. »A methodology for the development of distributed real-time control applications with focus on task allocation in heterogeneous systems«. In: *IEEE Transactions on Industrial Electronics.* Jahrgang 51, Heft 6 (2004), S. 1194–1207.

[Prieto-Díaz 1996] R. Prieto-Díaz. »Reuse as a New Paradigm for Software Development«. In: *Systematic Reuse: Issues in Initiating and Improving a Reuse Program.* Hrsg. von M. Sarshar. London: Springer, 1996, S. 1–13.

[Rausch & Broy 2008] A. Rausch und M. Broy. »Das V-Modell XT Grundlagen«. In: *Das V-Modell XT: Grundlagen, Methodik und Anwendungen.* Hrsg. von R. Höhn und S. Höppner. eXamen.press. Berlin Heidelberg: Springer, 2008, S. 1–27.

[Rezagholi 1995] M. Rezagholi. »Programm zur schrittweisen Ausrichtung der Softwareerstellung auf Wiederverwendung«. In: *Softwaretechnik – Trends.* Jahrgang 15, Heft 4 (1995), S. 38–43.

[Rupp et al. 2007] C. Rupp, S. Queins und B. Zengler. UML *2 Glasklar.* 3. Aufl. München Wien: Carl Hanser, 2007.

[Sandén 2001] B. Sandén. »A Design Pattern for State Machines and Concurrent Activities«. In: *Reliable Software Technologies – Ada-Europe 2001.* Hrsg. von D. Craeynest und A. Strohmeier. Lecture Notes in Computer Science. Berlin Heidelberg: Springer, 2001, S. 203–214.

[Sanz & Zalewski 2003] R. Sanz und J. Zalewski. »Pattern-based control systems engineering«. In: *IEEE Control Systems.* Jahrgang 23, Heft 3 (2003), S. 43–60.

[Scherff et al. 1999] B. Scherff, E. Haese und H. R. Wenzek. *Feldbussysteme in der Praxis – Ein Leitfaden für den Anwender.* Berlin Heidelberg: Springer, 1999.

[Schmidtberger 2008] T. Schmidtberger. »Wissensbasierte Auswertung von Anlagen-Planungsdaten für die Unterstützung des Prozessleittechnik-Ingenieurs – Anwendung einer rollenbasierten Mustersuche«. VDI, Düsseldorf, Fortschrittsberichte, Reihe 20, Rechnerunterstützte Verfahren, Nr. 415. Dissertation. Hamburg: Fakultät für Maschinenbau, Helmut-Schmidt-Universität, 2008.

[Schmitz et al. 2013] S. Schmitz, T. Jäger, S. Bleuel, L. Evertz, T. Nekolla, L. Urbas, S. Pech, U. Claussnitzer, T. Holm und M. Obst. »Anforderungen an die Automatisierungstechnik durch die Modularisierung verfahrenstechnischer Anlagen«. In: *IDA 2013 – Integrierte Digitale Anlagenplanung und -führung.* Frankfurt, Germany, 21. – 22. März 2013.

[Schnieder 1999] E. Schnieder. *Methoden der Automatisierung: Beschreibungsmittel, Modellkonzepte und Werkzeuge für Automatisierungssysteme mit 56 Tabellen.* 1. Aufl. Braunschweig Wiesbaden: Vieweg, 1999.

[Scholz 2005] P. Scholz. *Softwareentwicklung eingebetteter Systeme.* Berlin Heidelberg: Springer, 2005.

[Serna et al. 2011] F. Serna, C. Catalan, A. Blesa, J. M. Colom und J. M. Rams. »Predictive Maintenance Surveyor: Design Pattern for Machine Tools Control Software Applications«. In: *Tagungsband der 16. Tagung »IEEE International Conference on Emerging Technologies and Factory Automation« (ETFA).* Toulouse, France, 5. – 9. September 2011, S. 1–7.

[Stützle 2002] R. Stützle. »Wiederverwendung ohne Mythos: Empirisch fundierte Leitlinien für die Entwicklung wiederverwendbarer Software«. Dissertation. München: Fakultät für Informatik, Technische Universität München, 2002.

[Sünder et al. 2006] C. Sünder, A. Zoitl, J. H. Christensen, V. Vyatkin, R. W. Brennan, A. Valentini, L. Ferrarini, T. Strasser, J. L. Martinez-Lastra und F. Auinger. »Usability and Interoperability of IEC 61 499 based distributed automation systems«. In: *Tagungsband der 4. Tagung »IEEE International Conference on Industrial Informatics« (INDIN).* Singapore, 16. – 18. August 2006, S. 31–37.

[Szlachcic 2006] E. Szlachcic. »Fault Tolerant Topological Design for Computer Networks«. In: *Tagungsband der 1. Tagung »IEEE International Conference on Dependability of Computer Systems« (DepCos-RELCOMEX).* Szklarska Poreba, Poland, 25. – 27. Mai 2006, S. 150–159.

[Tauchnitz 2013] T. Tauchnitz. »Integriertes Engineering – wann, wenn nicht jetzt!« In: *atp-edition – Automatisierungstechnische Praxis.* Heft 1–2 (2013), S. 46–53.

[Thramboulidis 2006] K. Thramboulidis. »IEC 61 499 in factory automation«. In: *Advances in Computer, Information, and Systems Sciences, and Engineering.* Hrsg. von K. Elleithy, T. Sobh, A. Mahmood, M. Iskander und M. Karim. Dordrecht: Springer, 2006, S. 115–124.

[Thramboulidis 2008] K. Thramboulidis. »Challenges in the Development of Mechatronic Systems: The Mechatronic Component«. In: *Tagungsband der 13. Tagung »IEEE International Conference on Emerging Technologies and Factory Automation« (ETFA).* Hamburg, Germany, 15. – 18. September 2008, S. 624–631.

[Traub & Schraft 1999] A. Traub und R. D. Schraft. »An object-oriented Realtime Framework for distributed control systems«. In: *Tagungsband der Tagung »IEEE International Conference on Robotics and Automation« (ICRA).* Detroit, USA, 10. – 15. Mai 1999, S. 3315–3121.

[Vogel-Heuser 2003] B. Vogel-Heuser. *Systems Software Engineering – Angewandte Methoden des Systementwurfs für Ingenieure.* München: Oldenbourg Industrieverlag, 2003.

[Vogel-Heuser et al. 2013] B. Vogel-Heuser, C. Diedrich, A. Fay und P. Göhner. »Anforderungen an das Software-Engineering in der Automatisierungstechnik«. In: *Tagungsband der Konferenz »Software Engineering« (SE).* Aachen, Germany, 26. Februar – 1. März 2013.

[Vogel-Heuser & Bayrak 2012] B. Vogel-Heuser und G. Bayrak. »Ergebnisse der Agenda Cyber-Physical Systems für das Szenario smart factory«. In: *Tagungsband »Automation 2012«.* Baden-Baden, Germany, 13. – 14. Juni 2012.

[Vogel-Heuser & Witsch 2011] B. Vogel-Heuser und M. Witsch. *Erhöhte Verfügbarkeit und transparente Produktion.* Tagungsband Automation Symposium 2011. Kassel: University Press, 2011.

[von Aspern 2009] J. von Aspern. *SPS Grundlagen: Aufbau, Programmierung, Simulation, Internet und Sicherheit.* 2. Aufl. Heidelberg: Hüthig, 2009.

[Wagner & Göhner 2006] T. Wagner und P. Göhner. »Flexible Automatisierungssysteme mit Agenten«. In: *GMA-Fachtagung auf dem VDE-Kongress.* Aachen, Germany, 24. Oktober 2006.

[Wagner 2008] T. Wagner. »Agentenunterstütztes Engineering von Automatisierungsanlagen«. In: *atp-edition – Automatisierungstechnische Praxis.* Heft 4 (2008), S. 68–75.

[Wang & Chang 2008] C. S. Wang und C. T. Chang. »Integrated Genetic Algorithm and Goal Programming for Network Topology Design Problem With Multiple Objectives and Multiple Criteria«. In: *IEEE Transactions on Networking.* Jahrgang 16, Heft 3 (2008), S. 680–690.

[Weidemann & Drath 2010] D. Weidemann und R. Drath. »Einleitung«. In: *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA.* Hrsg. von R. Drath. VDI-Buch. Berlin Heidelberg: Springer, 2010, S. 1–44.

[Wenger 2010] W. Wenger. *Multikriterielle Tourenplanung.* Wiesbaden: Gabler GWV Fachverlag, 2010.

[Witsch et al. 2008] D. Witsch, A. Wannagat und B. Vogel-Heuser. »Entwurf wiederverwendbarer Steuerungssoftware mit Objektorientierung und UML«. In: *atp-edition – Automatisierungstechnische Praxis.* Heft 5 (2008), S. 54–60.

[Wurmus & Wagner 2000] H. Wurmus und B. Wagner. »IEC 61 499 konforme Beschreibung verteilter Steuerungen mit Petri-Netzen«. In: *Fachtagung »Verteilte Automatisierung«.* Magdeburg, Germany, 22. – 23. März 2000.

[Xia et al. 2003] F. Xia, Z. Wang und Y. Sun. »A design pattern for holonic manufacturing system in the IEC 61 499-based model-view-controller framework«. In: *Tagungsband der 1. Tagung »IEEE International Conference on Industrial Informatics« (INDIN).* Banff, Canada, 20. – 24. August 2003, S. 233–246.

[Xu et al. 2006] L. Xu, H. Ziv, T. A. Alspaugh und D. J. Richardson. »An architectural pattern for non-functional dependability requirements«. In: *Journal of Systems and Software.* Jahrgang 79, Heft 10 (2006), S. 1370–1378.

[Zeller & Prehofer 2013] M. Zeller und C. Prehofer. »Modeling and efficient solving of extra-functional properties for adaptation in networked embedded real-time systems«. In: *Journal of Systems Architecture.* 59.10 (2013), S. 1067–1082.

[Zou & Pavlovski 2006] J. Zou und C. J. Pavlovski. »Modeling Architectural Non Functional Requirements: From Use Case to Control Case«. In: *Tagungsband der Tagung »IEEE International Conference on Requirements Engineering« (RE).* Shanghai, China, 24. – 26. Oktober 2006, S. 315–322.

Referenzierte Normen, Richtlinien und Empfehlungen

[DIN 66 253-3 1989] *DIN 66 253-3: Programmiersprache PEARL – Teil 3: Mehrrechner-PEARL.* Deutsches Institut für Normung. 1989.

[IEC 60 848 2013] *IEC 60 848: GRAFCET specification language for sequential function charts.* International Electronical Commission. 2013.

[IEC 61 131-3 2003] *IEC 61 131-3: Speicherprogrammierbare Steuerungen – Teil 3: Programmiersprachen.* International Electronical Commission. 2003.

[IEC 61 131-3 2013] *IEC 61 131-3: Programmable controllers – Part 3: Programming language.* International Electronical Commission. 2013.

[IEC 61 499-1 2006] *IEC 61 499-1: Funktionsbausteine für industrielle Leitsysteme – Teil 1: Architektur.* International Electronical Commission. 2006.

[ISO 13 850 2014] *E DIN EN ISO 13 850:2014: Sicherheit von Maschinen – Not-Halt – Gestaltungsleitsätze.* International Organization for Standardization. 2014.

[ISO/IEC 15 909-1 2004] *ISO/IEC 15 909-1: Software and system engineering – High-level Petri nets – Part 1: Concepts, definitions and graphical notation.* International Organization for Standardization. 2004.

[ISO/IEC 25 010 2011] *ISO/IEC 25 010: Systems and software engineering – Systems and software Quality Reqirements and Evaluation (SQuaRE) – System and software quality model.* International Organization for Standardization. 2011.

[ISO/IEC 42 010 2011] *ISO/IEC 42 010: Systems and software engineering – Architecture description.* International Organization for Standardization. 2011.

[ISO/IEC 9126 2002] *ISO/IEC 9126: Software engineering – Product quality – Part 1: Quality model.* International Organization for Standardization. 2002.

[NA 35 2003] *Namur Arbeitsblatt 35: Abwicklung von PLT-Projekten.* Namur. 2003.

[VDI 2206 2004] *VDI Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme.* Verein Deutscher Ingenieure. 2004.

[VDI 36 95-1 2010] *VDI Richtlinie 3695-1: Engineering von Anlagen: Evaluieren und Optimieren des Engineerings – Grundlagen und Vorgehensweise.* Verein Deutscher Ingenieure. 2010.

[VDI 3633 2000] *VDI Richtlinie 3633: Simulation von Logistik-, Materialfluss- und Produktionssystemen – Grundlagen.* Verein Deutscher Ingenieure. 2000.

[VDI 3694 2008] *VDI Richtlinie 3694: Lastenheft/Pflichtenheft für den Einsatz von Automatisierungssystemen.* Verein Deutscher Ingenieure. 2008.

[VDI/VDE 2653-1 2010] *VDI/VDE Richtlinie 2653-1: Agentensysteme in der Automatisierungstechnik – Grundlagen.* Verein Deutscher Ingenieure. 2010.

Referenzierte Internetquellen

[Automation 2012] *VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik.*
URL: http://www.vdi.de/en/presse/details-english-press-releases/deutsche-automatisierer-haben-komplexe-aufgaben-im-griff/.
Zugriff am: 26. Mai 2015.

[autosar] *AUTOSAR – General Requirements on Basic Software Modules.*
URL: http://www.autosar.org/fileadmin/files/releases/4-0/software-architecture/general/standard/AUTOSAR_SRS_BSWGeneral.pdf.
Zugriff am: 26. Mai 2015.

[Industrie 4.0] *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0.*
URL: http://www.forschungsunion.de/pdf/industrie_4_0_abschlussbericht.pdf.
Zugriff am: 26. Mai 2015.

[omg] *Object Management Group.*
URL: http://www.omg.org/.
Zugriff am: 26. Mai 2015.

[sysml] *SysML Open Source Specification Project.*
URL: http://www.sysml.org/.
Zugriff am: 26. Mai 2015.

[uml] *UML: Object Management Group.*
URL: http://www.uml.org/.
Zugriff am: 26. Mai 2015.

[V-Modell xTa] *Das V-Modell xt.*
URL: http://www.cio.bund.de/DE/Architekturen-und-Standards/V-Modell-XT/vmodell_xt_node.html.
Zugriff am: 26. Mai 2015.

[V-Modell xTb] *Teil 5: V-Modell-Referenz Produkte.*
URL: http://download.4soft.de/v-modell-xt-bund/releases/1.0/html/14794f684e963e8.html.
Zugriff am: 26. Mai 2015.

Veröffentlichungen der Verfasserin

[Barth et al. 2012a] M. Barth, R. Drath, A. Fay, F. Zimmer und K. Eckert. »Evaluation of the openness of automation tools for interoperability in engineering tool chains«. In: *Tagungsband der 17. Tagung »IEEE, International Conference on Emerging Technologies and Factory Automation« (ETFA).* Krakau, Polen, 17. – 21. September 2012, S. 1–8.

[Barth et al. 2012b] A. Fay, R. Drath, M. Barth, F. Zimmer und K. Eckert. »Bewertung der Fähigkeit von Engineering-Werkzeugen zur Interoperabilität mit Hilfe einer Offenheitsmetrik«. In: *Tagungsband »Automation 2012«.* Baden-Baden, Germany, 13. – 14. Juni 2012, S. 1–14.

[Eckert et al. 2011] K. Eckert, T. Frank, T. Hadlich, A. Fay, B. Vogel-Heuser und C. Diedrich. »Typical Automation Functions and Their Distribution in Automation Systems«. In: *Tagungsband der 16. Tagung »IEEE, International Conference on Emerging Technologies and Factory Automation« (ETFA).* Toulouse, France, 5. – 9. September 2011, S. 1–8.

[Eckert et al. 2012] K. Eckert, T. Hadlich, T. Frank, A. Fay, C. Diedrich und B. Vogel-Heuser. »Design Patterns for Distributed Automation Systems with Consideration of Non-Functional Requirements«. In: *Tagungsband der 17. Tagung »IEEE, International Conference on Emerging Technologies and Factory Automation« (ETFA).* Krakau, Polen, 17. – 21. September 2012, S. 1–8.

[Fay et al. 2015] A. Fay, B. Vogel-Heuser, T. Frank, K. Eckert, T. Hadlich und C. Diedrich. »Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns«. In: *Journal of Systems and Software.* 101 (2015), S. 221–235.

[Frank et al. 2011] T. Frank, M. Merz, K. Eckert, T. Hadlich, B. Vogel-Heuser, A. Fay und C. Diedrich. »Dealing with non-functional requirements in distributed control systems engineering«. In: *Tagungsband der 16. Tagung »IEEE, International Conference on Emerging Technologies and Factory Automation« (ETFA).* Toulouse, France, 5. – 9. September 2011, S. 1–4.

[Frank et al. 2012a] T. Frank, K. Eckert, T. Hadlich, A. Fay, C. Diedrich und B. Vogel-Heuser. »Workflow and decision support for the design of distributed automation systems«. In: *Tagungsband der 10. Tagung »IEEE, International Conference on Industrial Informatics« (INDIN).* Beijing, China, 25. – 27. Juli 2012, S. 293–299.

[Frank et al. 2012b] T. Frank, T. Hadlich, K. Eckert, C. Diedrich und B. Vogel-Heuser. »Erweiterung des V-Modells® für den Entwurf von verteilten Automatisierungssystemen«. In: *Tagungsband »EKA 2012 – Entwurf komplexer Automatisierungssysteme«.* Magdeburg, Deutschland, 8. – 10. Mai 2012, S. 159–169.

[Frank et al. 2012c] T. Frank, T. Hadlich, K. Eckert, A. Fay, C. Diedrich und B. Vogel-Heuser. »Using contact points to integrate discipline spanning real-time requirements in modeling Networked Automation Systems for manufacturing systems«. In: *Tagungsband der 8. Tagung »IEEE, International Conference on Automation Science and Engineering« (CASE).* Seoul, Korea, 20. – 24. August 2012, S. 812–817.

[Frank et al. 2013a] T. Frank, K. Eckert, T. Hadlich, A. Fay, C. Diedrich und B. Vogel-Heuser. »Erweiterung des V-Modells® für den Entwurf von verteilten Automatisierungssystemen«. In: *at – Automatisierungstechnik.* Jahrgang 61, Heft 2 (2013), S. 79–91.

[Hadlich et al. 2011] T. Hadlich, C. Diedrich, K. Eckert, T. Frank, A. Fay und B. Vogel-Heuser. »Common communication model for distributed automation systems«. In: *Tagungsband der 9. Tagung »IEEE, International Conference on Industrial Informatics« (INDIN).* Lisbon, Portugal, 26. – 29. Juli 2011, S. 131–136.

[Hadlich et al. 2012] T. Hadlich, S. Höme, C. Diedrich, K. Eckert, T. Frank, A. Fay und B. Vogel-Heuser. »Time as non-functional requirement in distributed control systems«. In: *Tagungsband der 17. Tagung »IEEE, International Conference on Emerging Technologies and Factory Automation« (ETFA).* Krakau, Polen, 17. – 21. September 2012, S. 1–4.

Lebenslauf

Angaben zur Person	
Nachname(n) / Vorname(n)	Karin Eckert
Adresse(n)	Tulpenweg 10, 71106 Magstadt
Telefon	(0 71 59) 4 96 20 18 Mobil: 01 77-4 78 64 88
E-Mail	eckert.karin@gmx.net
Staatsangehörigkeit(en)	Deutsch
Geburtsdatum	1986-11-26
Geschlecht	weiblich
Schulbildung	
08/1993–07/1997	Grundschule in Richen
09/1997–07/1998	Hauptschule in Eppingen
09/1998–07/2003	Realschule in Eppingen Abschluss: Realschulabschluss
09/2003–07/2005	Berufskolleg I und II der Andreas-Schneider-Schule in Heilbronn Abschluss: Fachhochschulreife
Studium	
10/2005–07/2008	Studium der Wirtschaftsinformatik an der Hochschule Furtwangen Abschluss: Bachelor of Science
10/2008–09/2010	Studium der Information Systems an der Hochschule Pforzheim Abschluss: Master of Science
Beruf	
10/2010–12/2013	Wissenschaftliche Mitarbeiterin an der Professur für Automatisierungstechnik der Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg
Seit 1/2014	Software Engineer bei Thales Transportation Systems GmbH, Ditzingen