Methode zur Entwicklung und für den Test der Komponentenmodelle für die virtuelle Inbetriebnahme

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur

(Dr.-Ing.)

von **Dipl.-Ing. Zheng Liu**

geb. am 20.06.1984 in Hunan, China

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter:

Prof. Dr.-Ing. Christian Diedrich

Prof. Dr.-Ing. h. c. Michael Weyrich

Promotionskolloquium am 28. Juli 2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kurzfassung

Die steigende Nachfrage nach immer individuelleren Produkten und die damit einhergehende Forderung der Flexibilität der Anlagen sind für den Anlagenbetreiber eine Herausforderung. Er muss innerhalb kürzester die Anlagen umzurüsten und in Betrieb zu nehmen. Dabei spielen die Absicherung und der Test der Steuerungsprogramme eine wichtige Rolle, die heutzutage während der Inbetriebnahmephase noch sehr viel Zeit in Anspruch nimmt. Diesem Problem wird mit dem Simulationsverfahren „virtuelle Inbetriebnahme“ begegnet, mit dem eine vorgezogene Optimierung und Validierung der Steuerungsprogramme an einem digitalen Modell möglich ist. Zwar wird die virtuelle Inbetriebnahme bereits punktuell eingesetzt, allerdings stehen einem flächendeckenden Produktiveinsatz ein zu hoher Modellierungsaufwand des digitalen Modells der Anlage und eine unzureichende Integration in den Gesamtprozess gegenüber.

Innerhalb dieser Arbeit wird der Fokus auf das Verhaltensmodell der Komponenten der Anlage gelegt. Um den Aufwand zur Generierung des Komponentenmodells einzudämmen und deren Wiederverwendbarkeit zu erhöhen, wird ein neues Vorgehensmodell vorgeschlagen, welches die Modellentwicklung in die Komponenten- und Anlagenebene einteilt. Somit können Komponentenmodelle in der Zukunft von den Herstellern zusammen mit den Komponenten geliefert werden. Dieser Unternehmenübergreifende Prozess setzt jedoch voraus, dass die Komponentenmodelle in einem systemneutralen Format dargestellt werden, welches in dieser Arbeit vorgeschlagen wird. Um die Qualität der Komponentenmodelle zu steigern, wird eine Testmethode erarbeitet. Diese erhöht die Testabdeckung, in dem nicht nur die fehlerfreie Verhalten sondern auch die Fehlerverhalten einer Komponente getestet werden. Außerdem sinkt der Aufwand der Testausführung, indem der Testprozess werkzeugunterstützt ausgeführt wird. Zur Validierung der Konzepte werden zwei Fallstudien durchgeführt. Diese zeigen auch, das die erforschten Methodik praktisch umgesetzt werden kann.

Abstract

The increasing demand for more and more individual products and the associated requirement for flexibility of the systems are a challenge for the system operator. He has to convert the systems and put them into operation within a very short time. The securing and testing of the control programs play an important role, which still takes a lot of time during the commissioning phase today. This problem is countered with the simulation process „virtual Commissioning “, with which an early optimization and validation of the control programs on a digital model is possible. virtual commissioning is already being used selectively, but widespread productive use is hindered by excessive modeling effort for the digital model of the plant and insufficient integration into the overall process.

Within this thesis the focus is placed on the behavior model of the components of the plant. In order to reduce the effort involved in generating the component model and to increase its reusability, a new procedural model is proposed which divides the model development into the component and plant level. Thus, in the future, component models can be supplied by the manufacturers together with the components. However, this cross-company process assumes that the component models are presented in a system-neutral format, which is proposed in this thesis. In order to increase the quality of the component models, a test method is developed. This increases the test coverage in that not only the error-free but also the error behavior of a component are tested. In addition, the effort involved in the test execution is reduced because the test process is carried out with the aid of tools. Two case studies will be carried out to validate the concepts. These also show that the researched methodology can be implemented in practice.

1 Einleitung

1.1 Motivation und Ziel der Arbeit

Die auf Hardware-in-the-Loop basierte Methode der virtuellen Inbetriebnahme (VIBN) stellt ein wichtiges Hilfsmittel dar, um Steuerungssoftware zu testen, zu optimieren und somit die Inbetriebnahmezeit zu verkürzen. Eine validierte Komponentenmodellbibliothek ist die Voraussetzung für die Erstellung des Anlagenmodells und somit Schlüsselfaktor für den Erfolg eines Virtuellen-Inbetriebnahme-Projektes.

Die Verkürzung der Inbetriebnahmezeit wird dadurch erreicht, dass die Software parallel zum Aufbau oder Betrieb der Produktionsstrecke entwickelt und getestet werden kann. Somit ist es im Idealfall möglich, die fertige Steuerungssoftware direkt einzuspielen, die Parameter anzupassen und die Anlage in Betrieb zu nehmen. Der Testaufwand an der Produktionsanlage wird somit deutlich kürzer.

Viele Methoden der (semi-)automatischen Erstellung der Simulationsmodelle setzen eine Komponentenbibliothek voraus. Durch Instanziierung der Bausteine der Komponentenbibliothek ist eine automatisierte Generierung eines Simulationsmodells der ganzen Anlage realisierbar [MW09] [Mey14]. Trotz vieler Diskussionen ist eine automatisierte Modellgenerierung nur für Komponenten möglich, die eine einfache Funktionalität besitzen. Die Automatisierung erfolgt meistens durch Übernahme, Erarbeitung und Transformation von Engineeringdaten [Bar11].

In vielen Anlagen gibt es jedoch auch Komponenten, die komplexere technische Funktionen besitzen, die sich nicht nur auf das zeitliche und logische Verhalten reduzieren lassen. Beispielsweise verfügt ein Frequenzumrichter neben herkömmlichen Antriebssteuerungsfunktionen auch über administrative Aufgaben beispielsweise die Ansteuerung von verschiedenen Betriebsmodi. Für die Modellierung derartiger Komponenten ist ein detailliertes Know-how von der Komponente notwendig.

Bis dato liegt der Schwerpunkt der Komponentenmodellierung auf der Abbildung der fehlerfreien Verhalten, um die gewünschten Funktionen der jeweiligen Komponenten gegenüber der Steuerung zu testen. Zum Test der Störmeldung und Störbehandlungsfunktion in der Steuerung werden meist die Störverhalten während der virtuellen Inbetriebnahme durch

die Testingenieur manuell erzeugt. Dies muss manchmal durch Änderung der internen Struktur eines Komponentenmodells erfolgen.

Die Zielsetzung der vorliegenden Arbeit liegt darin, neuartige Methode zur Entwicklung und dem Test der Komponentenmodelle für die virtuelle Inbetriebnahme zu finden. Durch ein neues Vorgehensmodell soll ein unternehmenübergreifender Erstellungsprozess erzielt werden, der die organisatorische und technische Umsetzung der Verhaltensmodelle in der virtuellen Inbetriebnahme verbessern kann. Außerdem soll eine Testmethodik erarbeitet werden, um die Modellqualität zu steigern, die das Fehlerverhalten der Komponenten einschließt sowie den Aufwand der Testdurchführung reduziert.

1.2 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in neun Kapitel. In Kapitel 2 werden die für diese Arbeit relevanten Grundlagen und Begriffe erörtert. Zwei unterschiedliche Sichten zur Modellierung einer Anlage werden gezeigt. Die Definition und das Grundprinzip von virtueller Inbetriebnahme werden beleuchtet. Diese stellen die Grundgerüst für die späteren Teile dieser Dissertation dar.

Kapitel 3 befasst sich mit dem aktuellen Stand der Wissenschaft und Technik. Zu diesem Zweck wird zunächst auf verschiedene Verfahren zur Simulationsmodellierung eingegangen. Anschließend werden Ansätze zur Entwicklung des Simulationsmodells vorgestellt und bewertet. Vorhandene Methoden zur Reduzierung des Aufwands in den Modellierungsprozess werden gegenübergestellt und evaluiert. Damit ein Modell in geeigneten Detaillierungsgrad zu erstellen ist, sind viele Möglichkeiten in der Forschung vorhanden, die die Modellierungstiefe einstufen. Schließlich werden etablierte Methoden zur Validierung des Simulationsmodells eingeführt. Aus den obigen Forschungsergebnissen werden vier Defizite identifiziert, die eine weitere Anwendung der virtuellen Inbetriebnahme behindern. Diese vier Defizite werden als Forschungsbedarf für den restlichen Teil dieser Dissertation behandelt, aus denen Lösungskonzepte herausgearbeitet werden, die wiederum als Gliederung von Kapitel 4 bis 9 zu betrachten sind.

In Kapitel 4 wird ein modifiziertes Vorgehensmodell eingeführt, das auf dem klassischen V-Modell beruht. Aufbauend auf diesem Vorgehensmodell wird ein neues Vorgehensmodell vorgeschlagen, das die Entwicklung des Simulationsmodells in zwei Schritte - Komponentenebene und Anlagenebene - einteilt. Damit wird eine effiziente Umsetzung des unternehmenübergreifenden Lebenszyklus möglich. Dieses Vorgehensmodell dient auch als Dachkonzept für die weitere Entwicklung eines Anlagenmodells.

Kapitel 5 beleuchtet das Vorgehen der Verhaltensmodellierung für eine Komponente. Es wird gezeigt, dass es für unterschiedliche Anwendungsszenarien jeweils die passenden Modellierungsansätze - Black-Box-, Grey-Box- und White-Box-Modelle - einsetzen sollen. Zum Zweck der virtuellen Inbetriebnahme sind meistens Black-Box- und Grey-Box-Modelle zu bewerkstelligen. Neben dem Standardvorgehen für normale Komponenten (Abschnitt 5.2.1) wird in Abschnitt 5.2.2 gezeigt, wie die modernen Komponenten, die mit einer integrierten Steuerung ausgestattet sind, zu modellieren sind. Modellierung des Störverhaltens bildet ein weiteres Thema in Kapitel 5 ab. Dafür wird eine neue Herangehensweise vorgeschlagen, wie die Störszenarien systematisch und vollständig identifiziert werden können.

Das Vorgehensmodell aus Kapitel 4 schlägt vor, Komponentenmodelle von den Lieferanten zu erstellen. Die Problematik, wie diese Modelle bei den OEMs wieder in ein Anlagenmodell zusammenzubauen, wird in Kapitel 6 behandelt. Nach einem Vergleich verschiedener Lösungsmöglichkeiten wird ein neutrales Austauschformat, das auf erweitertem PLCopen XML basiert, vorgeschlagen. Auf dieser Basis wird am Ende des Kapitels ein angestrebter Entwicklungsprozess im Detail beschrieben.

Zur automatischen Durchführung des Tests und damit zur Verringerung des Aufwands der virtuellen Inbetriebnahme wird in Kapitel 7 ein Testframework dargestellt, welches die Testdurchführung erleichtern kann. Zusätzlich unterstützt das Testframework Fehlerinjektion, d.h. die Fehlerszenarien, die in Kapitel 5 konzeptionell eingeleitet wurde, lassen sich durch einen Fehlerinjektor im Simulationsmodell generieren. Darüber hinaus können Testergebnisse durch das Testframework bewertet werden.

In Kapitel 8 wurden zwei Fallstudien dargestellt, die die in vorherigen Kapiteln beschriebenen Lösungsansätze exemplarisch zeigen sollten. Fallstudie 1 zeigte auf, wie das Verhalten einer intelligenten Komponente - ein SEW-Frequenzumrichter - modelliert werden sollte. Im Anschluss wird das Verhaltensmodell mit dem in Kapitel 6 eingeführten neutralen Austauschformat dargestellt. Fallstudie 2 konzentriert sich auf die Modellierung und Testen des Verhaltens am Beispiel einer VEP-Lineareinheit. Sowohl das normale als auch fehlerbehaftete Verhalten wird abgebildet und getestet. Die Anwendung der in Kapitel 5 vorgestellten Methode zur Ableitung der Störszenarien wird hier auch beispielhaft beleuchtet.

In Kapitel 9 erfolgen schließlich eine kurze Zusammenfassung dieser Dissertation sowie ein Ausblick auf weitere Forschungsmöglichkeiten.

2 Grundlagen

2.1 Modelle und Systembeschreibung

2.1.1 Systembegriff

Der Begriff „System“ wird im Alltag und in der Literatur in unterschiedlichsten Bedeutungen und Definitionen verwendet.

Die Definition eines Systems hängt stark von der Sichtweise auf den Untersuchungsgegenstand ab. Nach [Gip13] stellt ein System „ein komplexes ‚Ganzes‘ dar, dass sich aus mehreren, mehr oder weniger deutlich unterscheidbaren Bestandteilen, den Objekten, zusammensetzt.“ Dieses „komplexe Ganzes“ kann materiell wie eine technische Einrichtung oder immateriell wie eine Datenbank sein.

F. Cellier hat in [CG13, S. 2f.] beschrieben: „The largest possible system of all is the universe [...] A system is characterized by the fact that we can say what belongs to it and what does not, and by the fact that we can specify how it interacts with its environment. System definitions can furthermore be hierarchical.“ [IEC09] definiert ein System als „Menge miteinander in Beziehung stehender Elemente, die in einem bestimmten Zusammenhang als Ganzes gesehen und als von ihrer Umgebung abgegrenzt betrachtet werden.“

Aus den beiden Definitionen ist zu sehen, dass eine Gegebenheit, die gegenüber der Umwelt eine Grenze hat, als ein System betrachtet werden kann.

Grundsätzlich kann es zwischen sozialen, ökologischen und technischen Systemen unterschieden werden. Als soziale und ökologische Systeme unterliegen z.B. Menschen, die die Systeme zugleich auch mitgestalten und verändern können [FMRS11]. Im Gegensatz dazu dienen technische Systeme zur Umsetzung eines definierten Eingangs von Stoff, Energie, oder Information in einen determinierten Ausgang dieser Größen [Rop75]. Demzufolge sind das Verhalten eines technischen Systems vorausbestimmbar. Im weiteren Teil dieser Dissertation wird nur das technische System betrachtet.

Das Innere eines Systems besteht aus Elementen, die „Eigenschaften besitzen und welche durch Relationen miteinander verknüpft sind“. Die gegenseitige Wechselwirkung zwischen gekoppelten Elementen bestimmen das Gesamtverhalten eines technischen Systems. Je nach Betrachtung kann ein Element selbst als ein Teilsystem aufgefasst werden, dadurch ist eine hierarchische Systemdarstellung möglich.

Die Abbildung der Systemgrenze schließt den Teil ein, der wegen bestimmten Aufgaben oder Zwecken von Interesse ist. Über die Systemgrenze hinweg kann ein System mit der Umwelt interagieren, um z.B. Materie, Energie und Informationen auszutauschen.

Folgende Charakteristika sind relevant für die Definition eines Systems:

- Systemgrenze
- Systemelemente
- Beziehungen zwischen Systemelementen
- Systemeingänge
- Systemausgänge

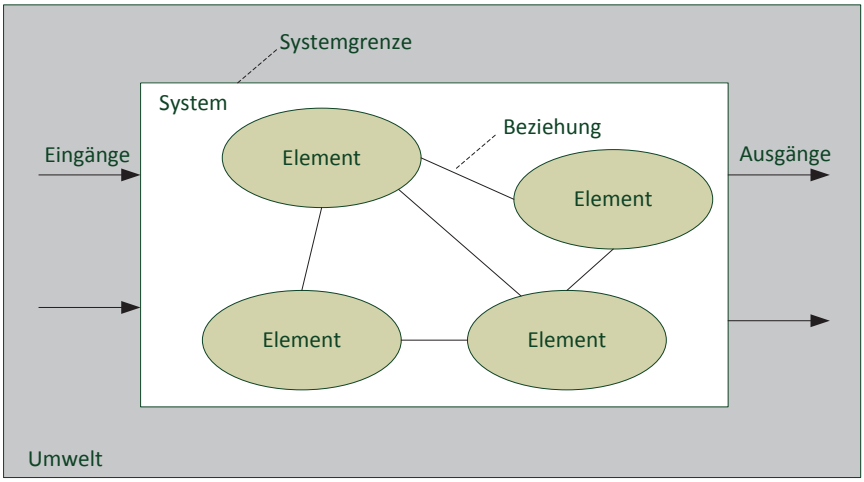


Abbildung 2.1: Aufbau eines Systems

2.1.2 Modell und Modellierung

Um den Aufbau und das Verhalten eines Systems zu beschreiben, wird ein Modell gebraucht. Ein Modell kann als eine vereinfachte Abstrahierung, die Eigenschaften eines realen Systems repräsentiert, verstanden werden. In [Sei03] wird Modell als eine Menge von Aussagen über ein untersuchtes System betrachtet. Nach [Sta73, S. 131-133f.] lässt sich ein Modell durch folgende drei Merkmale charakterisieren:

Abbildung: Modelle sind stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modell sein können.

Verkürzung: Modelle erfassen im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant scheinen.

Pragmatik: Modelle sind ihren Originalen nicht eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion a) für bestimmte - erkennende und/oder handelnde, modellbenutzende - Subjekte, b) innerhalb bestimmter Zeitintervalle und c) unter Einschränkung auf bestimmte gedankliche oder tätliche Operationen.

Es ist zu sehen, dass ein Modell nicht die Realität in allen Facetten sondern nur immer ein Teil davon wiedergeben.

Der Prozess um ein Modell abzubilden wird als „Modellierung“ bzw. „Modellbildung“ benannt. Aufgabe der Modellierung ist, entsprechend dem Modellzweck, ausgewählte Eigenschaften eines realen Systems abzubilden. Durch Modellierung wird „die Abbildung eines Systems oder Prozesses in ein anderes begriffliches oder gegenständliches System, das auf Grund der Anwendung bekannter Gesetzmäßigkeiten, einer Identifikation oder auch getroffener Annahmen gewonnen wird und das System oder den Prozess bezüglich ausgewählter Fragestellungen hinreichend genau abbildet“ [DIN94]. Kennzeichnend dafür ist nach [Bos13] „Vereinfachung“, „Zusammenfassung“, „Weglassen“ und „Abstraktion“. D.h. bei der Modellierung wird die Struktur oder das Verhalten eines originalen Systems durch ein Modell mit einem geringeren Detaillierungsgrad beschrieben.

Grundsätzlich kann ein System auf zwei unterschiedliche Arten modelliert werden. Eine Art ist die verhaltensbeschreibende Modellierung, wenn die interne Wirkstruktur des Originalsystems unbekannt ist. Da bei solchem System nur das Verhalten des Systems an der Systemgrenze bekannt ist, wird es als „Black-Box“ betrachtet. Die deskriptive Modellierung eines „Black-Box“-Systems basiert auf Beobachtung und Experiment am realen System, die die interne Realisierung nicht spezifiziert wird. Die Modellstruktur spiegelt in der Regel die Struktur des realen Systems nicht wider. Dabei werden nur die Stimuli am Eingang sowie die Reaktionen am Ausgang betrachtet [Bun63].

Wenn die Struktur eines Systems hingegen bekannt ist, lassen sich die kausalen Zusammenhänge innerhalb eines Systems mit physikalischen Gesetzen und mathematischen Formeln analytisch modellieren. Diese Methode ist als Glass- bzw. White-Box Methode bekannt.

Eine Mischform von Black-Box und White-Box, die als Grey-Box genannt wird, ist auch möglich, wenn nur Teile des Systems bekannt und nicht alle Wechselwirkungen zwischen den Systemelementen bekannt sind.

2.1.3 Simulation

Simulationen sind ein anerkanntes Hilfsmittel bei der Planung, Realisierung und beim Betrieb von technischen Systemen. Insbesondere wenn Untersuchungen an realen Systemen zu aufwendig, zu gefährlich, oder ethisch nicht vertretbar sind. Ursprünglich wurden Simulationen zur Absicherung der Planung benutzt. Im Laufe der Zeit erhielt die Simulation immer wieder neue Einsatzgebiete [VDI00]. Das Ziel der Simulation besteht darin, das Systemverständnis zu erhöhen, Optimierung am System durchzuführen oder dessen Verhalten vorherzusagen [Dro04].

Eine sehr allgemeine Definition von Simulation hat Shannon in [Sha75] beschrieben:

Simulation ist der Prozess der Modellbeschreibung eines realen Systems und des anschließenden Experimentierens mit dem Modell, um zu Erkenntnissen zu gelangen, die auf die Realität übertragbar sind.

Die VDI-Richtlinie 3633 definiert Simulation folgendermaßen [VDI00, S. 2f.]:

Simulation ist ein Verfahren zur Nachbildung eines Systems mit seinen dynamischen Prozessen mithilfe eines experimentierfähigen Modells, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind. Insbesondere werden die Prozesse über die Zeit entwickelt.

Folglich besteht die Aufgabe von Simulation in der Durchführung von Experimenten an einem Modell, um Kenntnisse über das reale System zu gewinnen. Ein Simulationsexperiment wird definiert als „die gezielte empirische Untersuchung des Verhaltens eines Modells durch wiederholte Simulationsläufe mit systematischer Parametervariation oder Strukturvariation“ [VDI00, S. 3f.]

Heute wird eine Simulation vermehrt rechnergestützt ausgeführt und in vielen Anwendungsgebieten als Computersimulation gleichgesetzt. In diesem Kontext ist Simulation nicht anders als „virtuelle Experimente auf dem Computer“ [BZBP09, S. 1f.]. Wobei die Simulation mit Hilfe eines Simulationsprogramms durchgeführt wird, bei dem die analytische Lösung des Modells numerisch approximiert wird. Die Durchführung einer Simulation am Rechner wird als Simulationsprozess bezeichnet.

Basierend auf oberem Verständnis wird Simulation in dieser Arbeit verstanden als:

Simulation ist ein computergestützter Modellierungsprozess zur Nachbildung der Struktur und des Verhaltens eines Systems und zur experimentellen Untersuchung an dem Modell.

2.2 Beschreibung der Produktions- und Fertigungsanlagen

2.2.1 Beschreibung der Anlagen aus mechatronischer Sicht

2.2.1.1 Mechatronik

Der Begriff *mechatronics* (deutsch Mechatronik) wurde erstmals in 1969 von Ko Kituchi, dem japanischen Präsidenten der YASKAWA Electric Corporation eingeführt [HTF96]. Dieses Kunstwort kombiniert die beiden Wörter Mechanics (Mechanik) und Electronics (Elektronik), welches die elektronische Funktionserweiterung mechanischer Komponenten bedeutet. Mit dem Aufkommen der Mikroprozessortechnik ist die Mechatronik um einen weiteren Teilaspekt, die Informationstechnik, erweitert worden.

In [Ise07] beschreibt Isermann Mechatronik als „ein interdisziplinäres Gebiet, bei dem folgende Disziplinen zusammenwirken: mechanische und mit ihnen gekoppelte Systeme, elektronische Systeme, Informationstechnik. Dabei ist das mechanische System im Hinblick auf die Funktionen dominierend. Es werden synergetische Effekte angestrebt, die mehr beinhalten als die reine Addition der Disziplinen“. Entscheidend ist hier die gezielte Ausnutzung des Synergieeffekts der einzelnen klassischen Technologie, um ein Gesamtoptimum des ganzen Systems zu erreichen.

Eine ähnliche Definition nach [VDI04] bezeichnet Mechatronik als „das synergetische Zusammenwirken der Fachdisziplinen Maschinenbau, Elektrotechnik und Informationstechnik beim Entwurf und der Herstellung industrieller Erzeugnisse sowie bei der Prozessgestaltung“. Demnach sind integrierter Entwurf und Fertigung der Produkte und Prozesse auch ein wesentliches Merkmal von Mechatronik.

Die in der Mechatronik involvierten Fachdisziplien werden in der Dissertation von [Kie07] um die Pneumatik und Hydraulik erweitert, die durch einen Einsatz im Karosseriebau geprägt sind.

Gemäß [VDI04] zeichnen sich mechatronische Systeme durch „funktionale und/oder räumliche Integration von Sensoren, Aktoren (auch Aktuatoren genannt), Informationsverarbeitung und einem Grundsystem aus.“ Durch Sensoren werden Zustandsgrößen des Systems erfasst. Die Informationsverarbeitung wertet diese Daten aus, um mittels Aktoren das System gezielt zu beeinflussen (vgl. Abbildung 2.2). Das Grundsystem setzt sich aus mechanischen, elektromechanischen, hydraulischen und/oder pneumatischen Komponenten zusammen.

Falls eine Benutzeroberfläche vorhanden ist, kann der Mensch mit dem System interagieren. Er kann auf interne Variablen zugreifen, sie wenn nötig verändern und so die Funktion individuell beeinflussen. Daneben kann eine Schnittstelle zu einer weiteren oder übergeordneten Informationsverarbeitung vorhanden sein.

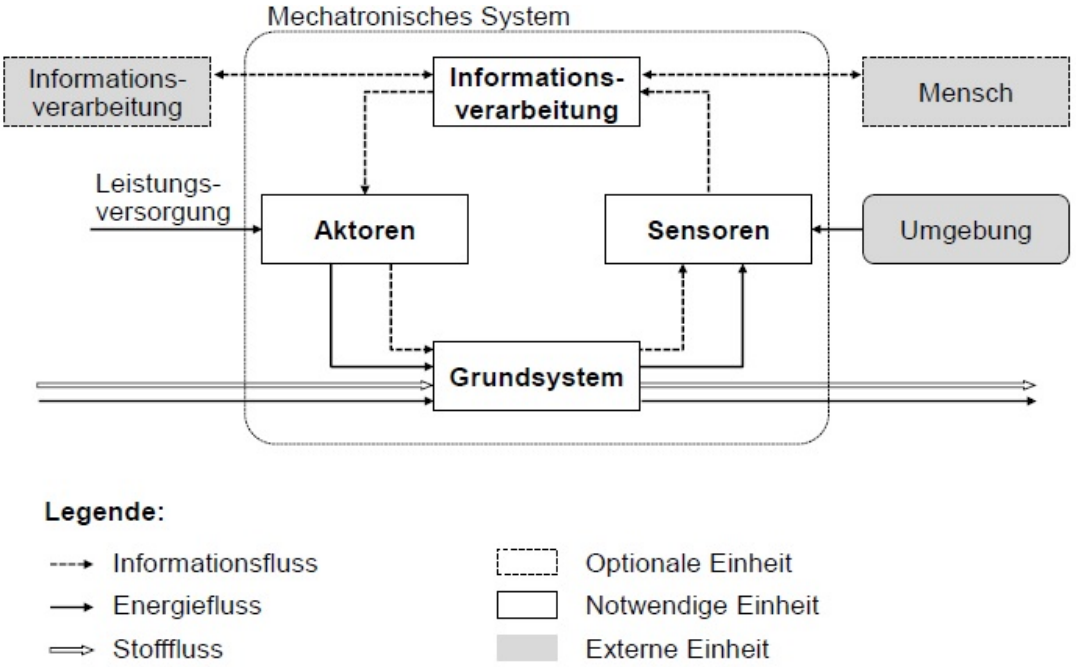


Abbildung 2.2: Grundstruktur eines mechatronischen Systems nach [VDI04]

Die Beziehungen bzw. Kopplungen zwischen den Bestandteilen eines mechatronischen Systems lassen sich in drei Gruppen aufteilen:

- Stoffflüsse: Beispiele für Stoffe in mechatronischen Systemen sind feste Körper, Gase oder Flüssigkeiten.
- Energieflüsse: Mechanische, thermische und elektrische Energie.
- Informationsflüsse: Unter Informationsflüssen werden in diesem Zusammenhang Messgrößen, Stellgrößen oder Daten verstanden.

2.2.1.2 Modellierung der Anlagen aus mechatronischer Sicht

Die Anwendungen von mechatronischen Systemen erstrecken sich auf viele industriellen Gebiete. Im Anlagenbau der Automobilindustrie, aufgrund des steigenden Einsatzes und der zunehmenden Vernetzung elektrischer, mechanischer und steuerungstechnischer Anlagenkomponenten, gewinnt das Thema „Mechatronik“ immer mehr Bedeutungen.

Als zentrale Einheit für die Informationsverarbeitung fungiert die speicherprogrammierbare Steuerung (SPS) (vgl. Abbildung 2.3). [IEC09] definiert SPS als „rechnergestützte

Leiteinrichtung oder rechnergestütztes System, deren oder dessen logischer Ablauf über eine direkt oder über Fernsteuerung angeschlossene Programmiereinrichtung, z.B. ein Bedienfeld, einen Hilfsrechner oder ein tragbares Terminal, veränderbar ist“.

Die Sensoren sind an die Eingänge der SPS geschaltet und erfassen und vermitteln Systemzustandsgrößen an die SPS. Typische Sensoren sind Lichtschrank, Endschalter, Näherungssensor usw. Anschließend verarbeitet SPS die Zustandsgröße auf Grundlage der geladenen Anwendungsprogramme, in den bestimmte Steuerungsalgorithmen programmiert sind. Die Ergebnisse werden als Stellgröße an die Aktoren weitergegeben, die die Zustände der Anlagen bzw. Maschine beeinflussen können. Beispiele für Aktoren sind Stellventil, Antriebe usw.

Die Aktoren und Sensoren sind mit der Anlage verbunden. Gemeinsam bilden sie ein mechatronisches System. Typische mechatronische Anlage ist eine Fertigungszelle im Automobilrohbau oder eine Materialflussstrecke. Zusammen mit Aktoren und Sensoren bildet das Anlagenverhalten in Abbildung 2.3 das Verhalten des ungesteuerten Systems. Die Prozessabläufe werden durch das Steuerungsprogramm definiert, indem die Aktionen des Verhaltens des ungesteuerten Systems durch die Abläufe in der SPS angestoßen werden. Die Zusammenwirkung des Steuerungsprogramms sowie des ungesteuerten Systemverhaltens bilden das gesteuerte Verhalten des gesamten Systems. Die Entwicklung einer mechatronischen Anlage lässt sich somit nach der mechatronischen Sicht in die Entwicklung der Steuerungstechnik und mechatronischen Anlagen einteilen.

2.2.2 Modellierung der Anlagen nach Merkmal-Sicht

Wie in Abschnitt 2.1.1 eingeführt, besteht ein System aus Teilsystemen und/oder Systemelementen. Dieses Prinzip lässt sich auch bei einer Anlage verwenden. Demnach werden komplexe Anlagen aus einzelnen, immer wieder verwendbaren Modulen zusammengesetzt. In [SD18] werden Methoden vorgestellt, wie mechatronische Komponenten bezüglich ihrer Funktionalität klassifiziert werden. Durch diesen Ansatz ist der Aufbau einer Anlage baukastenbasiert aus unterschiedlichen wiederverwendbaren kleineren Modulen möglich.

Jedes Modul weist gemeinschaftliche Eigenschaften - sowohl elektrisch als auch mechanisch und pneumatisch- auf und wird daher als mechatronische Komponente benannt. Eine Möglichkeit zur Beschreibung der Eigenschaften der mechatronischen Komponente ist das Merkmalmodell. Kerngedanke hier ist, alle Systemaspekte als Eigenschaften zu beschreiben (siehe auch [VSW+18]). Für die Modellbildung werden die Eigenschaften als Merkmale konkretisiert. Sowohl System als auch Systemelemente können mit Merkmalen beschrieben werden. Mit dem Merkmalmodell wird eine einheitliche Struktur zur formalen Beschreibung von Systemeigenschaften festgelegt. Der Ansatz beruht auf der Definition und Zuordnung

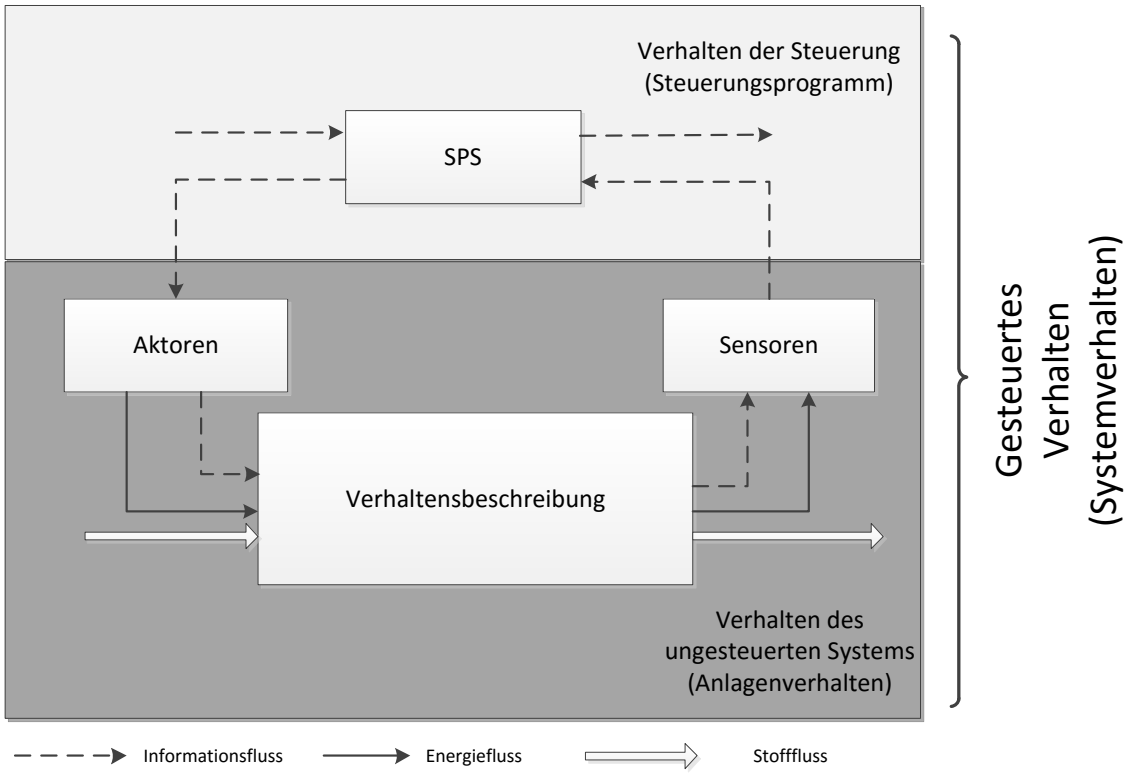


Abbildung 2.3: Grundstruktur eines mechatronischen Anlagensimulationsmodells

von Merkmalen zu Merkmalträgern. Das Merkmalmodell bildet die Eigenschaften der Systemelemente ab. Die Systemelemente werden als Merkmalträger bezeichnet, denen die Merkmale zugeordnet werden.

Nach [Had15] ist Merkmal „eine allgemein erkennbare Eigenschaft eines Betrachtungsgegenstandes, die zu einer Klassifizierung des Betrachtungsgegenstandes genutzt werden kann." Merkmale ermöglichen damit eine eindeutige Zuordnung von Begriff und denen quantifizierbaren Aussagen. Merkmale können in Beziehung mit einem oder mehreren Merkmalträgern stehen. Die Merkmalträger sind nicht nur Systeme und Systemelemente, die im Allgemeinen klassifiziert und je nach Systemart und Zweck des Systemmodells ausgeprägt sind. Bei einer mechatronischen Anlage können Merkmalträger Systemfunktionen repräsentieren, die jeweils die mechanischen, elektrischen und pneumatischen Aspekte eines Systems darstellen, u.a. sind Materialien, Produkte, Energien und Informationen (vgl. Abbildung 2.4).

Diese Betrachtungssicht entspricht auch dem Informationsmodell der Formalisierten Prozessbeschreibung (FPB, [VDI15a]), indem alle Kernelemente wie Produkte, Energien, Prozessoperatoren und technische Ressourcen als Objekt betrachtet werden, die mit Merkmalen zu beschreiben sind. Die Prozessoperatoren entsprechen den Systemfunktionen im Merkmalmodell, die von den technischen Ressourcen, d.h. von einem technischen System nach Merkmalmodell, bereitgestellt werden. Somit lässt sich eine Verbindung zwischen Funktionssicht eines Systemmodells und Merkmalmodells herstellen.

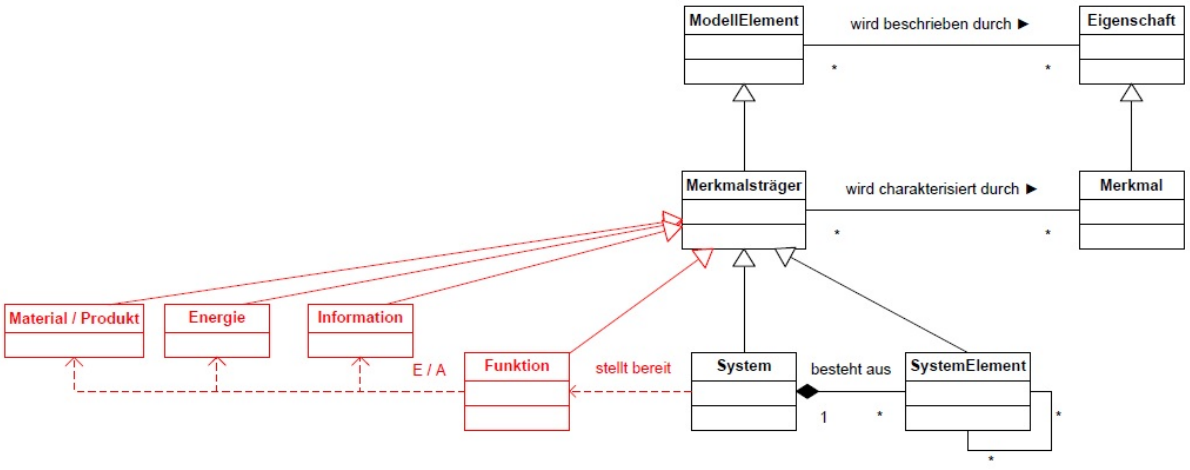


Abbildung 2.4: Zusammenhang zwischen Merkmal- und Systemmodell nach [Had15]

Mit der Darstellung nach Abbildung 2.4 werden alle Sichtweisen der technischen Ressourcen in ein Modell vereint, die verschiedene Ausprägungen der Anlage wiedergeben. Während zur Prüfung der mechanischen Funktionalität zum Beispiel Geometrie, Kinematik und auch Verhalten wichtig sind, benötigt man für die SPS-Programme die Aspekte Verhalten,

Kommunikation sowie Visualisierung und Steuerungstechnik (vgl. Abbildung 2.5). Aus diesen Aspekten werden die einzelnen Merkmale des Merkmalmodells abgeleitet.

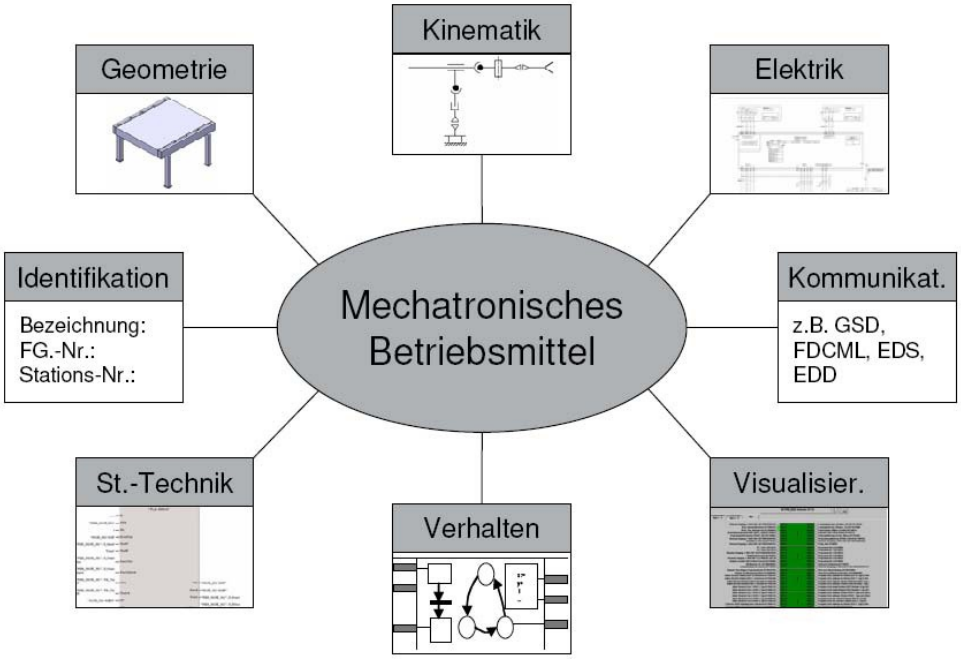


Abbildung 2.5: Aspekte mechatronischer Betriebsmittel [BDKS09]

2.2.3 Modellierung der Anlagen in Digitaler Fabrik

VDI-Richtlinie 4499 [VDI08] definiert die Digitale Fabrik als einen „[...] Oberbegriff für ein umfassendes Netzwerk von digitalen Modellen und Methoden unter anderem der Simulation und 3D-Visualisierung. Ihr Zweck ist die ganzheitliche Planung, Realisierung, Steuerung und laufende Verbesserung aller wesentlichen Fabrikprozesse und -ressourcen in Verbindung mit dem Produkt“.

In [IEC15] wird Digitale Fabrik als „digitale Repräsentation eines Produktionssystems“ definiert, welche ein Modell für die virtuelle Beschreibung einer realen Anlage darstellt. Durch die Digitale Fabrik werden Konzepte, sowie digitale Werkzeuge zur Planung, Modellierung und Simulation zur Verfügung gestellt. Um den Informationsaustausch zwischen Unternehmen, Engineering-Tools oder unterschiedlichen Phasen des Produktionslebenszyklus zu befähigen, wird in diesem Standard ein „Digital Factory Framework“ als Datenrepository eingeführt. Dieses Framework beschreibt neben Eigenschaften eines Elements, welches als elektronische Repräsentation eines Gerätes oder Assets von einem Produktionssystem verstanden wird, auch Beziehungen zwischen einzelnen Elementen.

Der große Vorteil des Konzepts der Digitalen Fabrik ist die Möglichkeit zu einer produktionsgerechten Produktauslegung, sowie zur optimalen Anpassung der zu entwickelnden Produktionssysteme und -prozesse an das Produkt.

In der Digitalen Fabrik werden sämtliche Produkt- und Fertigungsprozesse der realen Fabrik abgebildet. Das Konzept beinhaltet sowohl Softwaretools zur Verifizierung und Visualisierung von Produktionsprozessen, als auch Mitarbeiter und Prozesse, die zur Erstellung der zunächst virtuellen und später realen Produktion notwendig sind. Hier können neue Anlagen und Ressourcen für neue Produkte geplant und ausgelegt werden. Lange vor dem ersten Hardwaremodell wird mit Hilfe von virtuellen Entwicklungs- und Planungsmethoden, wie die Simulation, CAx-Methoden oder DMU sichergestellt, dass ein optimales Produkt in einer Fabrik mit optimalen Prozessen entsteht. Übrigens sollen die Realisierungs- und Anlaufphasen durch einen hohen Grad der Absicherung der Planungsergebnisse dramatisch verkürzt werden. Durch das gemeinsame Verständnis des Produkts und eine verbesserte Kommunikation zwischen den verschiedenen Teilbereichen bei der Planung und Realisierung sollen Entwicklungszeit durch größtmögliche Parallelisierung (engl. *Simultaneous Engineering*) und redundanzfreie Daten sowie die Qualität des Produkts optimiert werden.

Die Modelle der Digitalen Fabrik beruhen auf dem in der Produktionsdomäne etablierten PPR-Konzept (Produkte, Prozesse und Ressourcen). Nach PPR werden Produkte durch Prozesse hergestellt, die durch technische Ressourcen (Hardware, Software usw.) umgesetzt werden. Die Ressource stellen alle in der Ausführung eines Prozesses involvierten Hardware, Software usw. dar. Beruhend auf dem PPR-Konzept wird in der VDI/VDE-Richtlinie 3682 [VDI15a] eine formalisierte Prozessbeschreibung spezifiziert. Hierbei hat ein Prozess mehrere Eingangs- und Ausgangsgröße, die nicht nur Produkte sondern auch Energien und Informationen darstellen können. Einem Prozess kann eine technische Ressource zugeordnet werden, die dem technischen Prozess ausführt. Ein technischer Prozess lässt sich zudem in weitere Teilprozesse teilen.

Mit dem PPR-Konzept werden die Beziehungen zwischen den Prozessschritten, den Edukten und Produkten, den Eingangs- und Ausgangsenergien und -informationen und der technischen Ressource beschrieben. Diese Beschreibung bildet den Ausgangspunkt für das Automatisierungssystem. Ein Informationsmodell nach [HFMW17] berücksichtigt Abhängigkeiten zwischen Systemparametern, die nach PPR-Prinzip in drei Parametertypen zu unterscheiden sind: Produkt-, Prozess- und Ressourcenparameter. Anstatt die Zusammenhänge zwischen Systemkomponenten durch Wahrscheinlichkeiten bzw. Binärwerte zu beschreiben, bestrebt dieser Ansatz Abhängigkeiten zwischen den Systemparametern quantitativ mit Korrelation- oder Constraintsbeziehungen zu formulieren. Abhängigkeiten zwischen Elementen werden in einer Multiple Domain Matrix (MDM) gespeichert (Abbildung 2.6). Somit bietet MDM eine gute Möglichkeit, die statische Beziehung zwischen Systemkomponenten nach PPR-Sicht zu modellieren. [HFMW18] und [MHWF18] folgen diesen Ansatz weiter, um Modernisierungsmöglichkeiten des Automatisierungssystems zu generieren und zu bewerten.

	Process_Parameter_1	Process_Parameter_2	Process_Parameter_2	Process_Parameter_3	Resource_Parameter_1	Resource_Parameter_2	Resource_Parameter_3	Resource_Parameter_4	Product_Parameter_1	Product_Parameter_2	Product_Parameter_3
Process_Parameter_1					0		-			▲	
Process_Parameter_2					+		-		▼		
Process_Parameter_2						+			▼		
Process_Parameter_3						+		-			●
Resource_Parameter_1	0	+				▲+					
Resource_Parameter_1	0	+					▼+				
Resource_Parameter_2			+	+			▼-				
Resource_Parameter_2			+	+				▲-			
Resource_Parameter_3	-	-						●+			
Resource_Parameter_4				-		▼-			[<10]	/	/

Abbildung 2.6: Darstellung der Abhängigkeiten der Systemparameter mit Multiple Domain Matrix [HFMW17]

2.3 Virtuelle Inbetriebnahme

Die virtuelle Inbetriebnahme hat das Ziel, Aufgaben bei der Anlagenentwicklung in der Fertigungs- und Verfahrenstechnik in die frühen Engineeringphasen zu verlegen [BKHF09] [Kra07]. Dazu gehören z.B. die Validierung und der FAT (Factory Acceptance Test) [Web13], das betriebliche Training und die frühzeitige Absicherung und Optimierung der Planungsdaten. Dies ermöglicht es die Entwicklungs- und Inbetriebnahmezeiten zu verkürzen und Fehler früh zu erkennen.

Mit dem Konzept der Digitalen Fabrik ist es möglich, Produkte, Prozesse und Anlagen simulativ in Modelle abzubilden und darauf basierend die geplante Produktion virtuell am Rechner zu verbessern, um ein weitestgehend fehlerfreier Prozess für die reale Fabrik zu entwickeln. Damit ist Digitale Fabrik ein übergreifender Begriff, der die Fabrikplanung und reale Fabrik durch digitale Technik verknüpft.

2.3.1 Einordnung der virtuellen Inbetriebnahme in die Digitale Fabrik

In der Fertigungsindustrie ist die Digitale Fabrik seit vielen Jahren fester Bestandteil im Lebenszyklus einer Fertigungsanlage. Die Werkzeuge der Digitalen Fabrik erstrecken sich von der Layoutplanung und Konstruktion über eine Robotersimulation, bei der die Erreichbarkeit der Punkte und Gefahrenbereiche auf Kollision geprüft werden, bis zur Materialflusssimulation und Logistikplanung.

Grundsätzlich setzt sich der Fabrikbetrieb aus den Phasen Planung, Umsetzung, Betrieb in der realen Fabrik mit anschließender Bewertung des Ergebnisses [Wes03].

Die Planung beginnt mit der Konzeptplanung der Anlagen, in der ein grobes Konzept auf Basis der Produktentwicklung und der Vorgaben des Vertriebs angefertigt wird [SKB05]. In der anschließenden Detailplanungsphase lässt sich dieses Konzept detailliert ausarbeiten. Diese Phase kann nochmals in Mechanik-Konstruktion, Elektrokonstruktion und Erstellung der Steuerungsprogramme unterteilt werden (vgl. Abbildung 2.7).

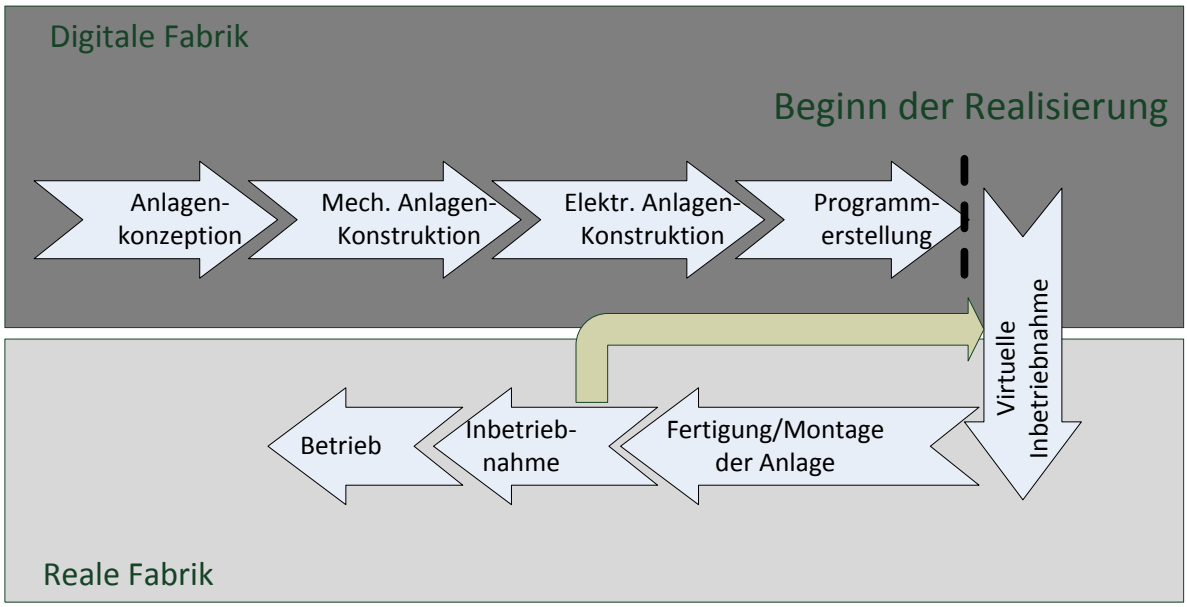


Abbildung 2.7: Einordnung der virtuellen Inbetriebnahme in die Digitale Fabrik

In einem traditionellen Fabrikbetrieb wird nach der Planung mit dem Aufbau der Anlage begonnen. Die benötigten Anlagenteile werden eingekauft oder speziell angefertigt und dann zur Gesamtanlage montiert.

Da Produktionsanlagen heute sehr komplex sind, muss das korrekte Zusammenspiel der verschiedenen Betriebsmittel vor dem Serienbetrieb in der Inbetriebnahmephase überprüft werden. In [DIN77] wird die Inbetriebnahme (IBN) beschrieben als „Bereitstellen einer Maschine oder eines vergleichbaren technischen Arbeitsmittels zur Nutzung“. Somit stellt die Inbetriebnahme „Funktionsbereitschaft und das funktionale Zusammenwirken der zuvor montierten Einzelkomponenten her und prüft die Korrektheit der Einzelfunktionen sowie deren funktionales Zusammenwirken“ [Zeu98]. Wurde die Anlage erfolgreich in Betrieb genommen, kann die Anlage in den Serienbetrieb überführt werden.

Die schrittweise Inbetriebnahme kann erst beginnen, wenn die Maschine fertig gestellt und elektrisch vollständig angeschlossen ist [Mew05]. Da bei der Inbetriebnahme die Anlagenkomponenten aller beteiligten Fachbereiche zum ersten Mal zusammengeschaltet werden, werden viele Fehler, die während der Planungsphase entstanden sind, erstens bei der Inbetriebnahme erkannt. Dies führt oft zu einer erheblichen zeitlichen Ausdehnung der Inbetriebnahme und hohen Kosten zur Fehlerbehebung. Aus Abbildung 2.8 ist zu

erkennen, je später ein Fehler entdeckt wird, desto größer ist der Aufwand zur Beseitigung des Fehlers.

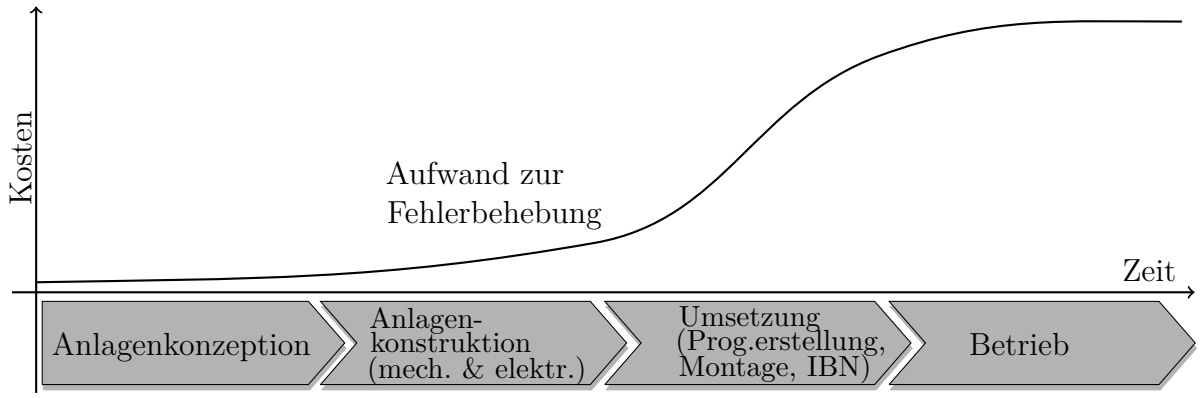


Abbildung 2.8: Kosten zur Fehlerbehebung in Abhängigkeit vom Zeitpunkt der Fehlererkennung (in Anlehnung an [Kie07])

Die virtuelle Inbetriebnahme, als ein Übergang zwischen der digitalen und realen Fabrik, ist der letzte Schritt vor der eigentlichen Inbetriebnahme, bei dem zum ersten Mal das Zusammenspiel aller Komponenten, mechanisch, elektrisch und informationstechnisch, überprüft werden kann. Sie stellt dadurch die Verknüpfung der Digitalen Fabrik mit der realen Fabrik dar. Die Grundidee der virtuellen Inbetriebnahme besteht in der Vorwegnahme des abschließenden Steuerungstestes anhand eines Simulationsmodells [Wün07]. Durch die zeitliche Vorverlegung des Funktionstests der Steuerungssoftware wird die Zeit für die reale Inbetriebnahme durch den Einsatz von Steuerungsprogrammen mit höherer Qualität verkürzt. Darüber hinaus bietet virtuelle Inbetriebnahme eine simulative Möglichkeit zur Überprüfung von möglichen Änderungen, die sich während der realen Inbetriebnahme ergeben haben, um eventuell gefährliche Situationen oder Kosten in der Realität zu vermeiden.

2.3.2 Definition und Grundprinzip

Die Inbetriebnahme hat nach Definition die Aufgabe, die Funktionsbereitschaft von montierten Anlagen zu sichern bzw. herzustellen. Nach [VDI15b] wird virtuelle Inbetriebnahme definiert als „*Inbetriebnahme, die das entwicklungsbegleitende Testen einzelner Komponenten und Teilfunktionen des Automatisierungssystems mithilfe von auf die jeweiligen Aufgabestellungen abgestimmten Simulationsmethoden und -modellen umfasst*“. Im engeren Sinne wird die virtuelle Inbetriebnahme als „*das Testen des gesamten Automatisierungssystems, unter Zuhilfenahme eines Anlagenmodells, vor der eigentlichen Inbetriebnahme*“.

In unterschied zur realen Inbetriebnahme, wird die realen existierende oder geplante Anlage durch das Modell simuliert. Mit dem erstellen Modell ist es möglich, das Zusammenspiel

von Maschine bzw. Anlage und dem dazu gehörigen Steuerungsprogramm mit Hilfe des Simulationsmodells zu testen. Damit wird die Funktionsbereitschaft von geplanten Anlagen vor der Anlaufphase zu sichern bzw. herzustellen und die Funktionalität zu optimieren.

		Anlage	
		virtuell	real
Steuerung	virtuell	Modell-in-the-Loop / Software-in-the-Loop	Reality-in-the-Loop
	real	Hardware in-the-Loop	Reale Inbetriebnahme / Betrieb

Abbildung 2.9: Unterschiedliche Kombinationen aus realen und virtuellen Komponenten

Abbildung 2.9 veranschaulicht vier Konstellationen - bestehend aus Steuerung und Anlage mit und ohne Simulation - für die virtuelle Inbetriebnahme.

Sind die Anlage und Steuerung real vorhanden, handelt es sich um eine reale Inbetriebnahme oder einen realen Betrieb. Das Verwenden einer virtuellen Steuerung mit einer realen Anlage wird als „Reality-in-the-Loop“ bezeichnet. Dies findet vor allem beim Test von Regelungssystemen Einsatz [FT06] und selten bei der virtuellen Inbetriebnahme.

Die Kombination aus realer Steuerung und virtueller Anlage wird als „Hardware-in-the-Loop“ (HiL) bezeichnet. Wenn sowohl die Steuerung als auch die Anlage virtuell vorhanden, handelt es sich um „Hardware-in-the-Loop“ (MiL) oder „Software-in-the-Loop“ (SiL).

Für die virtuelle Inbetriebnahme sind MiL, SiL sowie HiL relevant. Die drei Testmethoden unterscheiden sich in der Form der Implementierung des Steuerungscodes und das Vorhandsein der Steuerung (real oder emuliert) und werden im Folgenden näher erläutert [BPSH+15] [BK08b].

Model-in-the-Loop

Model-in-the-Loop (MiL) ist eine abstrakte Methode zum Testen des Steuerungsprogramms, die in frühen Projektphasen zur Validierung von prototypisch erstellten Steuerungsprogrammen oder zur Machbarkeitsstudie z.B. Taktzeitanalyse angewendet wird. Die Programme werden nicht nach einer klassischen Steuerungssprache wie z.B. IEC 61131-3 erstellt, sondern in der Modell-Sprache (z.B. auf Basis von Automaten) implementiert. Das erstellte Modell wird in einer simulierten Umgebung getestet und hinsichtlich der geforderten Funktionalität verifiziert. Typische Werkzeuge hier sind die grafisch orientierten Simulationswerkzeuge wie z.B. Matlab & Simulink. In der Regel wird das Anlagenmodell im gleichen Simulationswerkzeug abgebildet, in dem auch der Steuerungsgraph entwickelt

wurde. Demzufolge ist keine extra Kommunikationsschnittstelle zwischen Steuerung und Anlagenmodell notwendig. Da es ausreichend ist, wenn MiL die Übergangsbedingungen im Automatenmodell der Steuerung erfüllt, kann das Simulationsmodell hier einfach gestaltet werden.

Software-in-the-Loop

Bei dieser Simulationsmethode ist der Steuerungscode real in einer der Steuerungssprachen (z.B. IEC 61131-3) vorhanden. Der Steuerungscode wird auf einer emulierten Steuerung ausführt, die auf einem Rechner installiert und meist Teil des Engineering-Werkzeugs des Automatisierungssystems ist. Die Kommunikation mit dem Simulationsmodell erfolgt über virtuelle Kommunikationskanäle wie z.B. OPC, TCP/IP oder Shared Memory. Zweck des SiL-Testens ist es, mögliches Fehlverhalten nach einer Systemintegration zu identifizieren. Dieses Verfahren wird überwiegend in einem fortgeschrittenen Stadium der Projektierung eingesetzt, wenn die eigentliche Hardware, auf der das Steuerungsprogramm laufen soll, noch nicht vorhanden ist.

Hardware-in-the-Loop

Der grundlegende Unterschied zwischen SiL und Hardware-in-the-Loop (HiL) besteht darin, dass bei HiL der Steuerungscode auf einer realen Steuerung ausführt wird. Darüber hinaus wird die reale Kommunikationsschnittstelle (z.B. Profinet oder Profibus) eingesetzt. Dabei muss die Anlage in das Simulationswerkzeug mit geeigneten Schnittstellen modelliert werden. Jeder Teilnehmer am Bus wird von einem Emulator nachgeahmt. HiL eignet sich gut in späteren Projektierungsphasen, in dem die Materialbeschaffung bzw. der Anlagenaufbau durchgeführt wird.

	Steuerung	Kommunikations-peripherie	Anlage
Modell-in-the-Loop (MiL)	Modellsprache	keine	Vereinfachtes Simulationsmodell
Software-in-the-Loop (SiL)	Seriencode auf emulierter Steuerung	Virtuelle Kommunikation	Detailliertes Simulationsmodell
Hardware-in-the-Loop (HiL)	Ziel-Steuerungscode auf einer realen Steuerung	Reale Kommunikation	Detailliertes Simulationsmodell mit Schnittstellen

Tabelle 2.1: Testmethoden für die virtuelle Inbetriebnahme

Die drei Testmethoden für die virtuelle Inbetriebnahme werden nochmals in Tabelle 2.1 zusammengefasst. Außerdem gibt es bei HiL Möglichkeit, eine Hybride-Inbetriebnahme durchzuführen. Dabei werden sowohl reale als auch virtuelle Anlagenkomponenten über die reale Kommunikationsschnittstelle mit der Steuerung gekoppelt. Dabei können je nach Fortschritt der Inbetriebnahme zunehmend virtuelle durch reale Komponenten ersetzt werden (vgl. Abbildung 2.10).

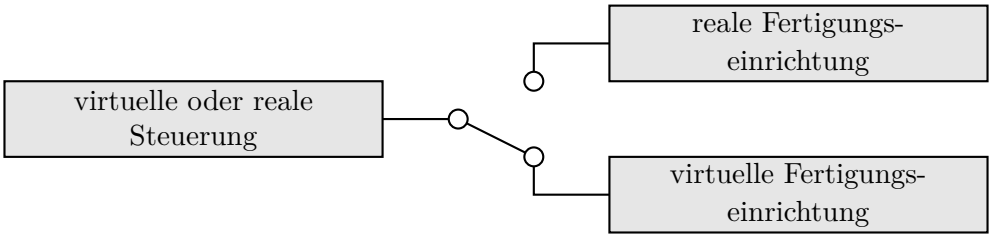


Abbildung 2.10: Prinzip der virtuellen Inbetriebnahme [VDI08]

Bei der virtuellen Inbetriebnahme wird eine real existierende oder geplante Anlage durch ein Simulationsmodell ersetzt. Wie schon in Abschnitt 2.2.1.2 eingeführt, wird im Simulationsmodell das Anlagenverhalten (ungesteuertes Verhalten des Systems) abgebildet.

Durch diesen Modellierungsvorgang werden die funktionalen Aspekte von Konstruktion, Elektrik und eingesetzten Geräte eindeutig in ihrer Struktur beschrieben und in ihrem Verhalten simuliert, so dass die zu entwickelnde Steuerungssoftware fortlaufend getestet und validiert werden kann. Das Simulationsanlagenmodell kann als strukturelle und funktionale Beschreibung für alle Lebenszyklusphasen genutzt werden. Beispiele sind:

- Erreichbarkeits- und Kollisionsuntersuchungen von bewegten Komponenten z.B. Robotern.
- Virtuelle Inbetriebnahme. Je nach geplanter Testtiefe erfolgt die Validierung der Steuerungssoftware für die reale Inbetriebnahme.
- Offline-Optimierungen der Betriebsparameter.
- Erleichterung der Fehlersuche während der Instandhaltung und
- Ermöglichung der Einweisungen bzw. des Trainings vom Bedienpersonal.

3 Stand der Forschung und Technik

3.1 Anlagensimulationsmodellierung

Dieser Abschnitt gibt einen kurzen Überblick über verschiedene Verfahren der Simulationsmodellierung der mechatronischen Komponenten.

3.1.1 Digital Mock-Up (DMU)

Unter dem Begriff Digital Mock-Up (DMU) versteht man ein virtuelles Versuchsmodell, das die Produktstruktur (Baugruppen, Einzelteile) und deren Geometrie repräsentiert, um Entwicklungskosten und -zeit während der Entwicklungsphase eines Produktes einzusparen. „Digital Mock-Ups sind entsprechend rechnergenerierte (digitale) Modelle oder auch Simulationen, die Aussehen und/oder Funktionalitäten zu entwickelnder Produkte weitestgehend originalgetreu darstellen.“ [KB16]. Der ursprüngliche Einsatz von DMU dient hauptsächlich die Produktanalyse. Auf Basis eines DMU können eine Vielzahl von Untersuchungen, wie Ein- und Ausbauuntersuchungen, Kollisionsprüfungen und Baubarkeitsprüfungen ausgeführt werden (vgl. [HKW08]). Schwerpunktmäßig findet DMU Anwendung im Bereich Produktentwicklung. Aufgrund ihres „Bindegliedcharakters“ zwischen Produkt- und Produktionsgestaltung findet DMU auch zunehmend Einsatz in der Produktionsplanung, z.B. die frühzeitige Absicherung der Produzierbarkeit von Bauteilen und Zusammenbauten unter Zuhilfenahme digitaler Betriebsmittel [Kie07]. In [Sch07] wird ein Simulationsmodell für VIBN erstellt, indem das Modell aus DMU kinematisiert wird. Dabei handelt es sich um die Definition von Achsen, von Bewegungsgrenzen und von Posen, also Bewegungsmustern.

3.1.2 Roboter-Offline-Programmierung

Die Programmierung eines Roboters unabhängig vom realen Robotersystem mithilfe eines Simulationsmodells nennt man Offline-Programmierung (OLP) [VDI07]. Im Vergleich zur „Online-Programmierung“, welche den Nachteil hat, dass die Programmierung direkt auf Robotersteuerung durchgeführt wird und somit die gleichzeitige Produktion blockiert wird, wird bei OLP die Roboterprogramme zunächst „Offline“ in einer Simulationsumgebung

entwickelt. Dadurch wird geholfen, die Kosten der Roboterinbetriebnahme zu senken. Im Anschluss an die 3D-gestützte Erstellung der Roboterprogramme müssen diese lediglich in die realen Robotersteuerungen transferiert und an die Abweichungen zwischen Modell und Realität angepasst werden [See99]. Nach [Ros11] besteht eine Simulationsumgebung zur Programmausführung für OLP aus Staarkörpersimulation, Verhaltenssimulation und 3D-Visualisierung.

3.1.3 Materialsimulationsmodell

Sowohl in der Auslegung einer neuen komplexen Produktionsanlage als auch in der Optimierung bestehenden Anlagen wird Simulation des Materialflusses als Hilfsmittel durchgesetzt, die es ermöglicht, die Wechselwirkungen von organisatorischen Strukturen, Investitionen in Maschinen und Transportsysteme, Durchlaufzeiten und Umlaufbeständen zu analysieren [GH99]. Plant Simulation ist eine Materialflusssimulationssoftware der Fa. Siemens zur Simulation komplexer diskreter Produktions- und Logistikprozesse [Mes17]. In dieser Software können Modelle von Produktionssystemen in Bezug auf Layout, Steuerungslogik und Ressourcendimensionierung ausgelegt werden, um die Eigenschaften der Systeme zu analysieren und die Leistung simulativ zu optimieren. Zahlreiche vorgefertigte Bausteiene sind in Plant Simulation beinhaltet für eine schnelle und effiziente Abbildung unterschiedlicher Fördermittel. [Pie17] beschreibt eine Möglichkeit, mit den Modellen aus Plant Simulation via die Schnittstelle OPC UA mit der SPS bzw. Soft-SPS z.B. PLCSIM zu koppeln. Dazu müssen Ein- und Ausgänge im Simulationsmodell angelegt und mit der Steuerung korrekt verknüpft werden.

3.1.4 Funktional Mockup Unit (FMU)

Das Functional Mock-up Interface (FMI) [MOD14] ist ein werkzeugunabhängiger Standard zur Unterstützung des Modellaustausches und der Co-Simulation.

Eine Instanz von FMI wird als Functional-Mockup-Unit (FMU) genannt [MOD14], die die Modellierung mit FMI ermöglicht. Mit Hilfe des FMIs wird eine Möglichkeit geschaffen, unterschiedliche FMUs unter demselben Framework zusammenzusetzen. Dadurch lassen sich Simulationsmodelle, die aus verschiedenen VIBN-Werkzeugen erstellt werden, für die Verhaltenssimulation einer Gesamtanlage miteinander koppeln und gemeinsam simulieren. Hierfür müssen die jeweiligen Simulationsmodelle in die FMUs überführt werden.

Jede FMU stellt eine komprimierte Datei mit der Erweiterung $.fmu$ dar und enthält sowohl das eigentliche Verhaltensmodell, das entweder als DLL oder als Quellcode weitergegeben werden kann, als auch das Modellbeschreibungsschema, das alle zur Simulation

benötigten Informationen wie z.B. Schnittstellenbeschreibung enthält. Die FMU-Datei lässt sich als Blackbox betrachten und ermöglicht knowhow-geschützten Austausch der Simulationsmodelle im Engineeringsprozess.

Der FMI-Standard unterstützt zwei Funktionsarten [BOA+12]. Bei „FMI für Modellaustausch“ ist nur die Modellfunktionalität in der Functional Mock-up Unit (FMU) hinterlegt. Für den Modellaustausch wird eine FMU von einem Simulationswerkzeug ohne Solver exportiert. Hier werden die Solver des Werkzeugs verwendet, in das die FMU importiert wurde. Im Gegensatz dazu wird im Fall „FMI für Co-Simulation“ auch ein Solver in FMU enthalten. Somit ist eine FMU in einem Werkzeug, das keinen Solver bereitstellt, auch alleine lauffähig.

Das Forschungsprojekt Avanti [ITEa] nutzt FMI-Standard zum Austausch von Simulationsmodellen zwischen verschiedenen Komponentenherstellern. Diese Modelle können wieder miteinander verschaltet werden via ein Co-Simulation-Netzwerk [SSD15]. In [HSSF17] wird gezeigt, dass ein FMU-Modell nicht nur in der Engineeringsphase sondern auch während des Betriebs Einsatz finden kann. Das Nachfolgeprojekt ENTOC [ITEb] folgt die Forschungsergebnisse von Avanti und versucht, standardisierte Beschreibungsformate und Schnittstellen zu definieren. Mit dem ausgewählten Datenaustauschformat AutomationML ist möglich, Informationen aus unterschiedlichen Fachdisziplinen sowie unterschiedlichen Engineeringsphasen, z.B. CAD-Daten, Kinematik, Geometrie, Verhaltensmodelle und Steuerungsprogramm gemeinschaftlich durch eine Komponentenbeschreibung zu verknüpfen. Dies bietet wiederum Möglichkeiten, FMUs aus der Komponentenbeschreibung direkt zu generieren.

Bei Komponentenbeschreibungen in AML können Informationen einer Komponente mit den entsprechenden CAD-Daten (inklusive Kinematik-Daten) oder Steuerungsprogrammen verknüpft werden.

3.1.5 Digitaler Zwilling

Der Begriff Digitaler Zwilling stellt ein virtuelles Abbild eines physischen Systems im Kontext des Cyber-physischen-Systems (CPS) dar [VDI17]. In einem CPS werden physischen Objekte und Prozesse aus verschiedenen Domänen mit virtuellen Objekten und Prozessen über eine Dateninfrastruktur vernetzt, um untereinander zu kommunizieren. Alle Objekte in einem CPS-Netzwerk können zur Laufzeit jederzeit in das Netzwerk eintreten und dieses wieder verlassen. Dies ist eines der Voraussetzungen für das Konzept „Plug-and-Produce “. In Anlehnung dazu wird in [JSW18] ein ähnliches Konzept „Plug-and-Simulate “ für den Digitalen Zwilling vorgeschlagen. Dafür sind eine Reihe von Anforderungen zu erfüllen [JJW17]:

- Eintritt und Verlassen der Modelle sowie Integration neuer Modelle in die Laufzeitumgebung
- Unterstützung der Komponentenmodellierung mit unterschiedlichen Simulationswerkzeugen
- dezentrierte Systemstruktur
- domänenunabhängig und lebenszyklusübergreifend

Nach [JSW18] soll das Multiagentensystem mit einer Co-Simulation-Struktur die obengenannten Anforderungen erfüllen. Dieser Ansatz koppelt verschiedene Simulationswerkzeuge dynamisch zusammen, in dem jede Komponente, die in einem Simulationswerkzeug modelliert wird, über eine Schnittstelle mit einem Softwareagenten verbunden ist. Die Agenten können zur Laufzeit in das Multiagentensystem eintreten und dieses wieder verlassen. Dies ermöglicht das dynamische Hinzufügen und Löschen von Modellen zur Laufzeit. Informations- und Datenaustausch zwischen Modellen werden durch die agentenbasierte Kommunikationsschnittstelle realisiert.

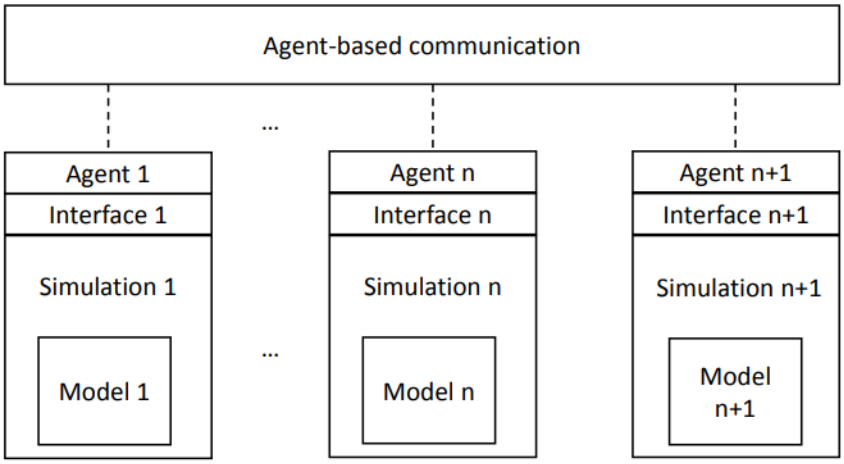


Abbildung 3.1: Multiagentenbasierte Co-Simulation nach [JSW18]

Der Digitale Zwilling grenzt sich von der virtuellen Inbetriebnahmen hauptsächlich dadurch ab, dass der Lebenszyklus der virtuellen Inbetriebnahme überwiegend mit der realen Inbetriebnahme endet (vgl. Abbildung 2.7), während der Digitaler Zwilling das Produktionssystem entlang des gesamten Lebenszyklus begleitet. Nach der Inbetriebnahmephase lässt sich der Digitaler Zwilling als ein virtuelles Pendant der physischen Instanziierung eines Produktionssystems betrachten, man spricht auch von betriebsparalleler Simulation. Dadurch können Aufgaben wie Echtzeitüberwachung, simulationsbasierte Optimierung, Systemneukonfiguration usw. ermöglicht werden [Zip21].

3.2 Modellentwicklungsprozess

Der größte Aufwand von virtueller Inbetriebnahme besteht in der aufwendigen Erstellung der Simulationsmodelle [Kie07] [Spi09]. In der Studie von [OBU15] wird gezeigt, dass über die Hälfte der Befragten den Aufwand für die Erstellung von Simulationsmodellen als hoch bis sehr hoch bewertet hat. Nach dem herkömmlichen Engineeringsprozess kann die Modellerstellung erstens anfangen, wenn notwendige Informationen in der Detaillierungsplanung entstanden sind. Dieser Modellierungsprozess findet parallel zum normalen Projektverlauf statt. Diese zusätzliche Aufwendung relativiert den Nutzen, der sich durch die virtuelle Inbetriebnahme ergibt. Um die Nutzungspotential auszuschöpfen, werden verschiedene Ansätze in der Industrie- und Forschungswelt probiert und analysiert.

3.2.1 Ableitung des Simulationsmodells aus Funktionsmodell

[Lin11] schlägt eine modellgetriebene Methode vor, dass Simulation frühzeitig in den Entwicklungslprozess integriert und entwicklungsbegleitend berücksichtigt werden soll. Dafür soll ein abstraktes Funktionsmodell zur Beschreibung der Maschinen zu Beginn des Entwicklungsprozesses eingeführt werden. In einem Funktionsmodell sind disziplinenübergreifende Informationen wie Steuerungs- und Maschinenverhalten enthalten, die im Laufe des Entwicklungsprozesses noch angereichert werden. Dieses Modell trägt auf einer Seite eine bessere Diskussionsbasis zwischen unterschiedlichen Fachbereichen bei, welches die Entwicklungszeiten beschleunigen wegen parallelisierter Entwicklungsschritte der verschiedenen Fachabteilungen. Auf anderer Seite lassen sich sowohl Modell für die virtuelle Inbetriebnahme als auch M-CAD und E-CAD-Dateien aus dem erweiterten Funktionsmodell ableiten.

Bewertung:

Zwar bietet das abstrakte Funktionsmodell eine Möglichkeit, alle Informationen entwicklungsbeleitend und disziplinenübergreifend in ein Modell zu integrieren und anschließend Simulationsmodell daraus abzuleiten. Diese modellgetriebene Entwicklung setzt eine Änderung des klassischen Engineeringsprozesses voraus, was ohne eine systematische Beschreibung durch ein Vorgehensmodell schwierig durchzusetzen ist. Zusätzlich besteht das ausgeleitete Simulationsmodell in [Lin11] aus C++ Code, das nur in einen eigenentwickelten Simulator ausführbar ist. Dies beschränkt die Nachvollziebarkeit und Lesbarkeit des Modells.

3.2.2 Entwicklung des Simulationsmodells in die Steuerungssoftware

Anstatt einer klaren Trennung des Steuerungs- und Simulationsteils werden in [Sei13] und [SS13] ein Ansatz vorgeschlagen, indem das Simulationssystem integriert in der Steuerung entwickelt wird. Für jede Komponente wird zusätzlich zu seinem Steuerungsfunktionsbaustein jeweils ein paralleler Simulationsbaustein in derselben Entwicklungsumgebung (Steuerungssoftware) entwickelt. Der Signalaustausch zwischen den beiden Bausteinen erfolgt z.B. über einen gemeinsamen Datenbaustein. Dieser Ansatz bietet den Vorteil, dass der Aufwand zur Erstellung des Simulationsmodells gering gehalten wird. Da es keine Einarbeitung für den Modellentwickler benötigt und die Kommunikation zwischen Steuerung und Simulation einfach zu konfigurieren ist. Zudem kann für jedes Feldgerät Ansteuer- und Simulationsprogramm in derselben Bibliotheksstruktur abgespeichert werden, die projekttechnisch einfach zu pflegen ist.

Bewertung:

Dieser Ansatz bietet die Vorteile, dass es keinen zusätzlichen Bedarf besteht für die Einarbeitung in eine Simulationsumgebung, da das Modell von demselben Programmierer erstellt werden kann, der die Steuerungsprogramme schreibt. Jedoch werden die Modellierungsumgebungen und Beschreibungsmitteln von der verwendeten SPS bestimmt, die die Modellierungsmöglichkeiten für die Simulationsmodelle beschränken kann. Der zusätzliche Simulationsteil muss nach der virtuellen Inbetriebnahme ausgeschaltet werden, was eine komplizierte Programmstruktur und eine zusätzliche Anforderung zur Versionierung bedeutet. Ein zusätzlicher Nachteil wäre, dass Denkfehler des Entwicklers in beide Modelle so eingebaut werden können, dass die Zusammenarbeit der beiden Teile trotzdem funktioniert. Die Denkfehler bleiben unentdeckt. Um dies zu vermeiden soll die Entwicklung der SPS-Programme und die Simulationsmodelle möglichst getrennt erfolgen.

3.2.3 Integration der Simulation in den Engineeringprozess

In [OWBU15] wird die Anwendung von Simulation in der Verfahrenstechnik in vier Gruppen zusammengefasst: „Design-Simulation“, „virtuelle Inbetriebnahme und simulationsbasiertes Engineering“, „Operator-Training“, „Betriebsbegleitende Entscheidungsunterstützung und Optimierung“. Da diese vier Anwendungsfällen bereits alle Phasen des Lebenszyklus einer Prozessanlage gedeckt haben, stellen die Autoren eine Vision dar, dass in der Zukunft Simulation systematisch genutzt und integraler Bestandteil der normalen Engineering- und Betriebsprozesse über den gesamten Lebenszyklus von Prozessanlagen sein soll.

Bewertung:

Der Ansatz nach [OWBU15] mit integrierter Simulation basiert auf einer Tool-Kette der Fa. Siemens (Comos, Simit etc.). Für einen allgemeinen Anwender kann jedoch sein, dass es verschiedene Softwarewerkzeuge für unterschiedliche Anwendungsfälle eingesetzt werde und jedes dieser Werkzeuge unterschiedliche Dateiformate für die Datenablage verwendet, kommt es häufig zu Problemen für den Datenaustausch zwischen den Werkzeugen. Dies macht eine durchgängige Nutzung von Simulation schwierig. In [FNMS12] wird ein auf AutomationML basierter Entwicklungsprozess dargestellt, der Integration unterschiedlicher Modelle zu einem Gesamtmodell ermöglicht.

3.3 Methoden zur Reduzierung des Aufwands in den Modellierungsprozess

3.3.1 Automatisierte Modellgenerierung

Die Idee einer automatischen Generierung von Simulationsmodellen resultiert aus der folgenden Analyse: die meisten Daten, die zur Simulation benötigt werden, werden schon während der Planungsphase erzeugt und in Planungsdatenbanken gespeichert. Infolgedessen könnte es angestrebt werden, das Simulationsmodell viel schneller und mit wenigen manuellen Tätigkeiten zu generieren. Unter dem Begriff der (teil-)automatischen Modellgenerierung werden im Simulationskontext Ansätze verstanden, bei denen ein Simulationsmodell nicht manuell mit den Modellierungswerkzeugen des gewählten Simulators erzeugt wird, sondern vielmehr über Schnittstellen und Algorithmen aus externen Datenquellen generiert wird [SBMS10].

Nach [SBMS10] werden drei Ansätze zur automatisierten Erstellung von Simulationsmodellen genannt:

- Parametrischer Ansatz: Diesem Ansatz liegt ein Bibliothekskonzept zu Grunde. Simulationsbausteine werden in Bibliotheken gespeichert und von dem Modellgenerator konfiguriert.
- Strukturbasierter Ansatz: Anfangspunkt hierbei sind Daten, in denen die Struktur des abzubildenden Systems beschrieben ist.
- Hybrid-wissensbasierter Ansatz: Diesem Ansatz liegen Verfahren der künstlichen Intelligenz (Expertensysteme, neuronale Netze) zu Grunde.

3.3.1.1 Semiautomatisierte Modellgenerierung

In [Spl96] wird davon ausgegangen, dass „eine automatische Übernahme vordefinierter Steuerungsstrategien aus einer Datenbank [...] unwahrscheinlich ist“. Steuerstrategien sind Regeln, die immer anwendungs- oder unternehmenbezogen aufgebaut sind und oft nicht in strukturierter Form, sondern nur als verbale Aussage vorliegen. Deshalb spricht [Spl96] eher von einer teilautomatischen Generierung von Simulationsmodellen. Die Methodik basiert auf eine systemneutrale Simulationsmodellbeschreibung. Diese wird aus verschiedenen Datenbanken und IT-Systemen anhand einer standardisierten Schnittstelle erzeugt. Der Simulant kann dann sein Wissen benutzen, um nur das Simulationsmodell zu ergänzen, insbesondere mit der Definition der Steuerstrategien. Die Methodik wurde in [Sel05] weiterentwickelt, um die Steuerstrategien ins Modell auch automatisch generieren zu können (vgl. Abbildung 3.2).

Die Methodik erfordert eine Menge von Schnittstellen. Z.B. sollen die Planungsdaten aus den Datenbanken anhand Visual Basic Programmen in MS Excel exportiert werden. Es ist auch zu konstatieren, dass „die Definition von Modellelementklassen die Generierung der Modelle erleichtern kann“. Eine aufwändige Erarbeitung an der Definition solcher Bibliothekselemente wird dann erforderlich.

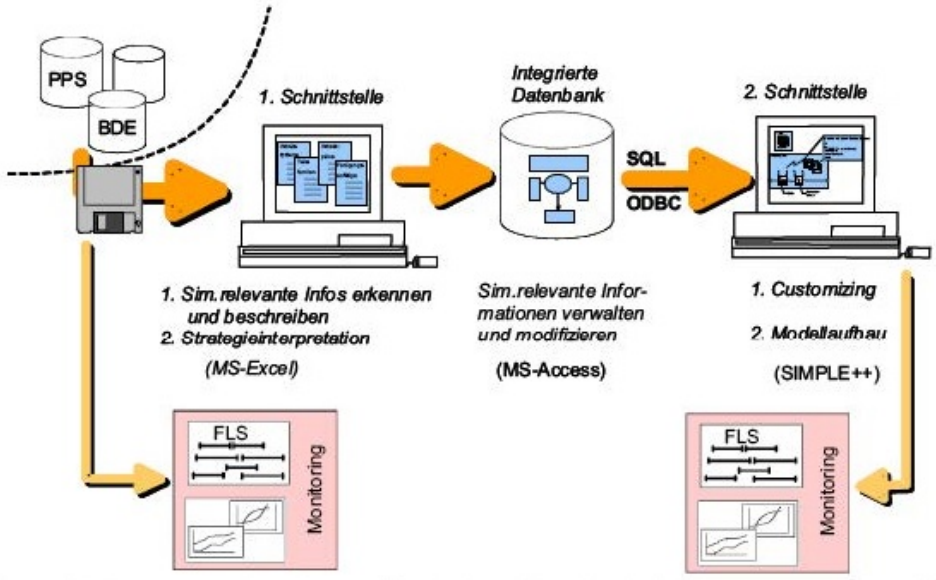


Abbildung 3.2: Struktur zur automatischen Simulationsmodellgenerierung [Sel05]

[Bar11] stellt drei Konzeptvarianten zur automatischen Modellgenerierung für die verfahrenstechnische Anlagen vor: K1) CAE-basierte Modellgenerierung mit Modelica, K2) PLS-basierte Modellgenerierung mit Excel und K3) Hybride Modellgenerierung. Datenquellen für K1 stellen die im CAE-System umgesetzte Anlagenplanung in Form des

Rohrleitungs- und Instrumentenfließbildes dar. Variante K2 siehe das PLS-Engineering-System als Datenquelle vor. K3 basiert auf der Kombination beider Datenquellen, deren Konsistenz hier jedoch zu erachten sind. Dabei ist die erste Variante objektorientiert, die eine vollständig automatische Generierung eines Simulationsmodells für Steuerungstests ermöglicht, indem die objektorientierte Planungsdaten in CAE-System zum objektorientierten Simulationsmodell transformiert werden. Die auf den Daten des PLS-Engineerings basierte Modellgenerierung ermöglicht die Instanziierung einzelner Objekte wie z.B. ein Ventil. Die Verbindung zwischen den Simulationsobjekten erfolgt nur manuell, um ein Prozessmodell der Anlage zu generieren.

3.3.2 Modellwiederverwendung

Ein Großteil der Kosten bei der virtuellen Inbetriebnahme entsteht bei der Erstellung der Verhaltensmodelle [BHH+09]. Dieser Aufwand kann unter Umständen so hoch sein, dass er den Nutzen übersteigt. Die Wiederverwendung von bereits erstellten Modellen sowie die Modellierung von Standardkomponenten und deren qualitative Absicherung im Vorfeld nehmen dabei eine entscheidende Rolle ein, um diesen Aufwand zu reduzieren.

[Tom98] schlägt eine komponentenbasierte Modellierung vor. Als Komponenten werden meist Baugruppen oder Geräte bezeichnet, die der Konstrukteur oder der Steuerungstechniker als geschlossenes System sieht. Die Eigenschaften sind meist aus den Produktunterlagen der Komponentenhersteller zu entnehmen. Komponenten können zu Subsystemen in Form von Funktionseinheiten zusammengefasst werden. Das Verhalten des Gesamtsystems entsteht durch Kombination mehrerer Funktionseinheiten.

[Xu03] überträgt Prinzipien aus der Softwareentwicklung auf die Modellierung für die maschinennahe Simulation. Als Ergebnis entstehen allgemeine Qualitätsanforderungen an eine Komponentenbibliothek. Dazu gehören

- Allgemeingültigkeit,
- Verständlichkeit und Strukturiertheit,
- Flexibilität und
- Validierbarkeit der Modelle.

In [Lin08] und [KOB09] wird empfohlen, dass Komponentenmodell als Bibliothekselemente von Lieferanten zur Verfügung gestellt werden soll. [SSD15] folgt dieser Idee und schlägt vor, dass ein sogenanntes MBM (Manufacturer Behaviour Model), d.h. Verhaltensmodell von Herstellern, von den Komponentenherstellern an die Kunden als Teil des Lieferumfangs mitgeliefert werden soll. MBMs sind für unterschiedliche Anwendungszwecke konzipiert und

können daher als Basis für weitere von Kunden erstellte Modelle (UBM: User Behaviour Model) dienen. UBM kann MBM adaptieren und erweitern mit zusätzlichen kundenspezifischen Funktionen, wie z.B. zusätzliche Signale für die Sicherheit oder Visualisierung. Hierbei ist die interne Struktur eines MBMs für Kunden nicht sichtbar und nicht änderbar, um das interne Knowhow des Komponentenherstellers zu schützen. In einem UBM kann ein oder mehrere MBMs eingepackt werden.

3.3.3 Modularisierung

Modularisierung beschreibt das Baukastenprinzip, nach dem ein Gesamtobjekt durch die Kombination mehrerer Einzelbausteine gebildet wird, wobei einzelne Bausteintypen auch mehrfach verwendet werden können.

Nach [Kah13] stellt das Ziel der Modularisierung die bessere Erweiterbarkeit und Wiederverwendbarkeit von Funktionalitäten dar. Ein Modul wird als eine logische oder physische Einheit mit abgegrenzten Aufgaben für ein Gesamtsystem beschrieben.

Die Modularisierung beschreibt ein Gesamtobjekt, das aus mehreren Einzelbausteinen besteht. Diese können mehrfach verwendet werden. Bei dem modularen Aufbau sind die einzelnen Bausteine leicht austauschbar und die komplexen Systeme werden besser verständlich.

Hierfür bietet es sich an, dass die kleinsten verwendeten Einheiten, wie beispielsweise Motoren, Frequenzumrichter oder Ventile, als fertige Bibliotheken modelliert werden. Diese Modellierung erfolgt zukünftig im Idealfall seitens der Hersteller, ähnlich wie dies schon mit GSD- oder CAD-Dateien erfolgt, dafür ist es lediglich notwendig, passende Austauschformate zu finden.

Nach [Wün07] lässt sich der Einsatz von virtueller Inbetriebnahme in drei Einsatzgebiete bzw. Abstraktionsebenen unterscheiden (vgl. Abbildung 3.3).

- Leitebene
- Zellebene
- Maschinenebene

Auf der Leitebene liegt das größte Abstraktionsniveau vor. Betrachtet wird auf dieser Ebene die Fördertechnik, welche einzelne (Fertigungs-) Zellen miteinander verbindet. Die einzelne Zelle ist nicht von Interesse und kann als Blackbox abgebildet werden. Der Fluss zwischen den Zellen wird durch eine den Zellen übergeordnete Steuerung koordiniert. Im Fokus stehen nicht die Einzeloperationen innerhalb der Zellen, sondern vielmehr die Koordination und Synchronisierung der verknüpfenden Fördertechnik.

Bei der Simulation oder Emulation auf Zellebene wird die Zelle steuernde SPS betrachtet. Diese ist der Steuerung der Leitebene untergeordnet und koordiniert die Fördertechnik, Fertigungsmaschinen und weitere Komponenten wie z. B. Zufuhrsysteme oder Spannvorrichtungen innerhalb einer Zelle. Fertigungsmaschinen wie Roboter besitzen häufig eine eigene Steuerung. Diese wird auf Zellebene nicht berücksichtigt. Äquivalent zu den Zellen auf der Leitebene werden die einzelnen Maschinen auf der Zellebene als Blackbox abgebildet.

Auf der Maschinenebene - der Modellierung mit dem höchsten Detaillierungsgrad - wird die Steuerung der einzelnen Maschinen innerhalb einer Zelle sowie die Koordination der Maschinen untereinander betrachtet. Die für die Steuerung von Robotern notwendigen Programme werden in der Regel mit Hilfe von Offline-Simulation entwickelt. Ein weitverbreitetes Werkzeug für diese Zwecke ist z. B. „Robcad“ der Fa. Siemens.

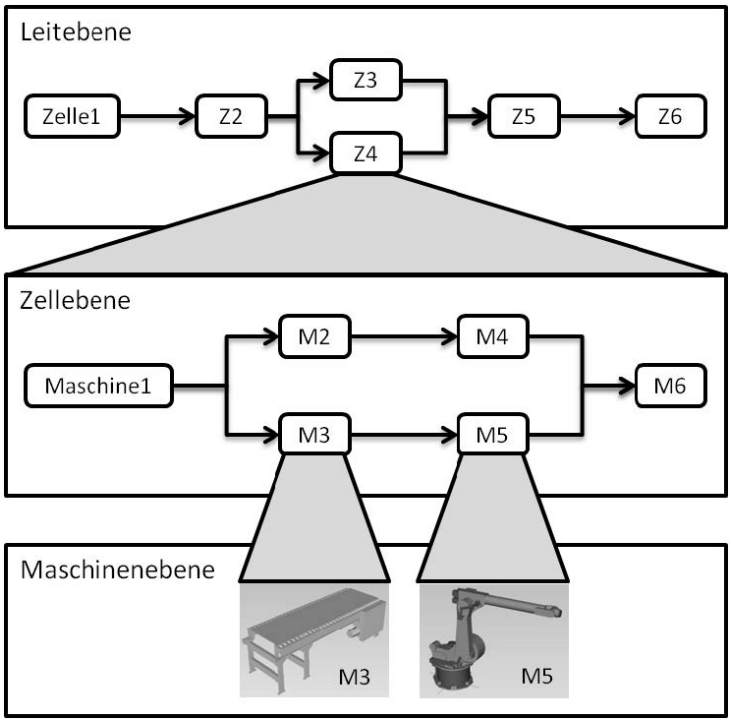


Abbildung 3.3: Abstraktionsebenen der Simulationsmodelle in der virtuellen Inbetriebnahme nach [Wün07]

In [Kuf12] wird die Modularisierung einer Montagemaschine in drei Ebene vorgesehen:

- Modul
- Komponente
- Bauteil und Baueinheit.

Dabei wird Modul als eine Gruppe von Komponenten, die unternehmensneutral und herstellerübergreifend zusammengesetzt sind. Eine Komponente besteht wiederum aus einer Baueinheit oder mehreren Bauteilen. Der Unterschied zwischen Bauteil und Baueinheit

besteht darin, dass Bauteil wie Ventile, Lampen oder Sensoren keine eigene Funktion erfüllen kann. Dagegen kann Baueinheit spezifische Funktionen realisieren und eine integrierte Regelung besitzen. Mit einem Modellierungstool lassen sich die Bauteile in der einfachen Modellierungstiefe automatisiert erstellen. Eine automatisierte Modellierung der Verhaltensmodelle für Baueinheiten ist dagegen nicht möglich. Die Autorin hat jedoch empfohlen, dass die durch den Lieferanten bereitgestellt werden sollen. Eine automatische Generierung der Maschinenmodelle aus Komponenten sowie die Anbindung an die Steuerung ist mit dem Modellierungstool ebenso realisiert.

Das Modularisierungskonzept in [PS16] basiert auf ähnlichem Prinzip wird jedoch mit anderen Terminologien beschrieben. Dabei wird Modularisierung als „Granularität“ bezeichnet. Nach der Auflösung des eingesetzten Anlagensimulationsmodells wird die Granularität in drei Ebenen hierarchisiert:

- grobgranular (Gesamtanlagen- / Funktionsgruppenebene)
- mittelgranular (Modulebene)
- feingranular (Komponentenebene)

Für jede granulare Ebene wird zwischen hoher und niedriger Detaillierung unterschieden. Wobei mit niedriger Detaillierung nur einzelne Teilaspekte eines realen Prozesses abgebildet werden können. Hohe Detaillierung erlaubt dagegen ein realistischeres Abbild, was jedoch auch zum höheren Modellierungsaufwand und Modellkomplexität führen kann.

Für virtuelle Inbetriebnahme sind nur mittel- bzw. feingranular aufgelöste Simulationsmodelle in Modul- oder Komponentenebene geeignet. Da grobgranulare Simulationsmodelle in hoher Detaillierung zu kompliziert und aufwendig sind, während in niedriger Detaillierung nicht aussagekräftig genug für spezifisches Verhalten einzelner Anlagen sein könnten.

3.4 Modellierungstiefe

Ein allgemeines Problem während der virtuellen Inbetriebnahme stellt die Frage nach der Modellierungstiefe der Modelle dar, die entscheidet, wie genau das Verhalten der zu modellierten Komponenten simuliert wird. Es ist leicht ersichtlich, dass ein zu weit vereinfachtes Modell die Realitätsnähe nicht mehr ausreichend gewährleisten kann. Dies wiederrum führt dazu, dass die Ergebnisse der virtuellen Inbetriebnahme nur bedingt nutzbar wären. Ein zu komplexes Modell führt zu einem erheblichen Modellierungsaufwand für die Modelle, was wiederrum den wirtschaftlichen Nutzen der virtuellen Inbetriebnahme übersteigen kann. Zum anderen besteht die Gefahr, dass heutige, handelsübliche Rechner eine echtzeitfähige Simulation dieser Modelle nicht gewährleisten könnten. Es sollte also

neben der Realitätstreue der virtuellen Inbetriebnahme stets deren Kosten/Aufwand-Verhältnis berücksichtigt werden. Die Herausforderung besteht darin, den „richtigen“ Detaillierungsgrad zu finden, der nicht kategorisch festgelegt werden kann, sondern vom jeweiligen Anwendungsgebiet und individuellen Anforderungen abhängt.

3.4.1 nach Abstraktionsgrad

In der VDI 2206 [VDI04] wird der Abstraktionsgrad nach der Darstellungsform der Modelle definiert, nach dem er in vier Ebenen unterteilt werden kann:

- **Topologisches Modell:** Im topologischen Modell wird die Topologie des Systems modelliert, sowohl die Anordnung einzelner funktionserfüllender Elemente als auch deren Verknüpfung untereinander. „Ein Element repräsentiert im Allgemeinen die drei Basisfunktionen kinematische Funktion (z.B. Gelenkanzahl...), dynamische Funktion (z.B. Bewegung von Massen unter Einwirkung von Kräften) und mechatronische Funktion (z.B. Regelung, Überwachung...).“
- **Physikalisches Modell:** Das physikalische Modell baut auf dem topologischen Modell auf und stellt dieses mit systemangepassten Größen wie Massen, Federn, Dämpfern, Widerständen, Spulen etc. nach. Das physikalische Modell beschreibt die Systemeigenschaften in domänenspezifischer Form.
- **Mathematisches Modell:** Das mathematische Modell bildet die Grundlage zur Verhaltensbeschreibung des Systems. Dabei wird das physikalische Modell in eine abstrakte, systemneutrale Form überführt (z.B. Differentialgleichungssystem). Die Modellierungstiefe kann hier z.B. durch Vernachlässigung von Störgrößen wie Reibung beeinflusst werden.
- **Numerisches Modell:** Das numerische Modell stellt eine algorithmisch behandelbare Aufarbeitung des mathematischen Modells dar. Das numerische Modell wird für die Berechnung parametrisiert, also mit konkreten Zahlenwerten belegt.

Bewertung:

Die vier Modellkategorien haben eine ansteigende Abstraktion nach der Reihenfolge. Es besteht jedoch keine eindeutige Verknüpfung zwischen Abstraktionsgrad und Modellierungstiefe. Ein Modell mit einer bestimmten Modellierungstiefe lässt sich mit unterschiedlichen Abstraktionsebenen abbilden. Dementsprechend ist ein Modell mit einem bestimmten Abstraktionsgrad wie beispielsweise mathematisches Modell sowohl in geringer Modellierungstiefe, z.B. durch Annäherungsmethode als auch in hoher Modellierungstiefe dargestellt werden.

3.4.2 nach Kufner

Nach [Kuf12] wird die Modellierungstiefe in fünf Stufen definiert, die in Tabelle 3.1 zusammengefasst wiedergegeben werden.

Modellierungstiefe	Inhalt des Abbildes
Logisches Abbild (kommunikationsfähig) (Modellierungstiefe I)	Abbild der Ein- und Ausgänge, die an der Maschine anliegen, sowie der logischen Zusammenhänge, die die Eingänge mit den Ausgängen verbinden.
Logisches Abbild mit Zeitverhalten (Modellierungstiefe II)	Aufbauend auf das logische Abbild wird in diesem Abbild ein Ausgang erst nach einer bestimmten Verzögerung gesetzt. Die Verzögerung entspricht der Aktionszeit der Komponenten.
Physikalisches Prinzipabbild (domänenübergreifend) (Modellierungstiefe III)	Darstellung der elementaren mechanischen, elektrischen u. hydraulischen Wirkketten der Maschine, d.h. die Komponenten werden in Anlehnung an die Inhalte einer Wirkstrukturskizze nach der VDI-Richtlinie 2222 festgehalten.
Physikalisches Konstruktionsabbild (domänenspezifisch) (Modellierungstiefe IV)	Erweiterung des physikalischen Prinzipabbildes durch domänenspezifisches Wissen. Genaue Beschreibung der physikalischen Effekte der Wirkkette anhand der Daten, welche bei der Konstruktion ermittelt wurden.
Physikalisches Zustandsabbild (mit Fehlermodellierung) (Modellierungstiefe V)	Annäherung des physikalischen Konstruktionsabbildes an reale, zu einem Zeitpunkt betrachtete Maschinen. Erfassung von Störgrößen durch fehlerhaft gefertigte Komponenten, eine fehlerbehaftete Montage sowie durch Verschleiß.

Tabelle 3.1: Definition der Modellierungstiefen nach [Kuf12]

Die Modellierungstiefe I und II sind geprägt durch die Abbildung des logischen Verhaltens zwischen Ein- und Ausgängen. Dabei wird die Modelllierungstiefe um ein Zeitverhalten, das auf Erfahrungen oder Schätzungen beruht, erweitert. Beginnend mit der Modellierungstiefe III, wird das Verhalten basierend auf physikalischen Wirkketten berechnet. Diese werden in Modellierungsstufe IV und V um physikalisch modellierte Fehler ergänzt.

Zusammenfassend werden in [Kuf12] zwei Szenarien betrachtet, die für die virtuelle Inbetriebnahme von Montagemaschinen einen besonders hohen Nutzen haben. Diese sind die Absicherung von Steuerungsprogrammen und die Optimierung der Taktzeiten.

Für die Optimierung von Taktzeiten eignen sich je nach Anforderung die Modellierungstiefen II-IV besonders gut. Näheres dazu ist in [Kuf12] zu finden.

Für die Absicherung der Steuerungsprogramme während der virtuelle Inbetriebnahme sind besonders die Modellierungstiefe I und II interessant. Bei der Modellierungstiefe I

ist ein Programmtest zwar möglich, der Testablauf wird allerdings erschwert und zeitabhängige Fehlerszenarien können nicht simuliert werden. Die Modellierungstiefe III und größer liefern oft kaum zusätzlichen Nutzen, da aus Sicht des Steuerungsprogramms die Berechnungsgrundlage der Prozesssignale irrelevant ist (physikalisches Modell oder Schätzung/ Erfahrungswert). Gleichzeitig steigt an dieser Stelle der Modellierungsaufwand stark an. Deshalb ist die Modellierungstiefe II vom Nutzen/Aufwand-Verhältnis häufig die geeignetste Lösung.

Bewertung:

Zwar wird die Modellierungstiefe hier in fünf auswählbare Stufen differenziert, hat die Autorin allerdings behauptet, dass für die meisten Anwendungen in der virtuellen Inbetriebnahme die Modellierungstiefe II bewerkstelligen soll. Lediglich für einige Anwendungen wie Berechnung eines Verfahrweges bzw. Ermittlung von Parametern werden höhere Detaillierungsstiefe benötigt. In der anschließenden Fallstudie in derselben Arbeit werden alle Modelle ebenso höchsten bis zur Modellierungstiefe II abgebildet. Demzufolge sind die Auswahlmöglichkeiten der Modellierungstiefe für die virtuelle Inbetriebnahme in der Praxis stark beschränkt.

3.4.3 nach Puntel-Schmidt

In [PS16] werden die Detaillierungsstufen in Anlehnung an die Sichtweise in der Verkehrssimulation definiert. Dort sind drei Detaillierungsstufen etabliert: Die makroskopische, mikroskopische und mesoskopische Modellierung [Hel13]. In Anlehnung an diese Sichtweise und Terminologie wird vier Detaillierungsstufen definiert.

- **Makroskopische Modelle** (einfache Totzeitmodelle): Modelle werden hier als Ganzes durch die Totzeit nachgebildet, die die Stellsignale in Rückmeldesignale umsetzt. Hierbei wird das zu simulierte Objekt als ein geschlossenes System unter Vernachlässigung des inneren Aufbaus betrachtet.
- **Mesoskopische/nicht dynamische Modelle** (einfache Geräte- und Kinematikmodelle): Die innere Struktur wird signalflussbasiert durch unterliegende abgebildet. Physikalisches Verhalten sowie komplexe Kinematik wie Beschleunigung werden bei der Modellbildung vernachlässigt.
- **Mesoskopische/dynamische Modelle** (Geräte und einfache Dynamikmodelle): Im Vergleich zur letzten Modellkategorie wird einfaches physikalisches Verhalten betrachtet. Z.B. die Bewegung eines Stückguts, die durch darauf wirkende Kraft hervorgerufen wird.

- **Mikroskopische Modelle** (Komplexe Geräte- und Dynamikmodelle): Diese Modelle sind in der Lage, komplexe freie Bewegungen von Stückgütern physikalisch abzubilden. Physikalische Verhalten sind z.B. Kollision mit Störkanten oder Verklemmung an Engpassstellen. Diese Modellkategorie hat die tiefste Detaillierungsstufe und stellt daher die höchste Anforderung an die Modellbildung. Sie ermöglichen ein detailliertes Testen der zu automatisierenden Funktion wegen der größten Realitätstreue.

Zusätzlich wird eine Domain-Mapping-Matrix (DMM) eingeführt, die für jede einzelne Unterkomponente Detaillierungsgrad bestimmen kann. Der dem oberen Modul zugewiesene Detaillierungsgrad für einen einzelnen Prozessschritt ergibt sich wiederum aus dem höchsten identifizierten Detaillierungsgrad der Komponenten innerhalb eines Moduls. Da einer einzelnen Komponente mehrere Prozessschritte zugeordnet werden können, wäre ein „dynamischer Detaillierungsgrad“ nach [FSGT15] sinnvoll, d.h. Umschaltung der Detaillierungsstufen für verschiedene Prozessschritte, um die Rechenzeit zu senken. Wegen der gewählten Konstellation aus Modelica-Modellen und HiL-Simulation hat der Autor jedoch gezeigt, dass derzeit kein technischer Lösungsweg bekannt ist.

Bewertung:

Die Klassifikation nach [PS16] weist einen starken materialflusstechnischen Hintergrund auf. Abgrenzungen zwischen einzelnen Detaillierungsstufen sind ausgeprägt durch die physikalische Eigenschaft von den betrachteten Stückgütern und Schüttgütern insbesondere in der mesoskopischen und mikroskopischen Stufe. Auswahl der Detaillierungsstufe für die einzelne Komponente setzt klare definierte Prozessschritte voraus. Dies beschränkt auch den obengenannten Ansatz Anwendungen in einen kontinuierlichen Prozess.

3.4.4 nach Schmüdderrich

Um die richtige Auswahl der Modellierungstiefe zu treffen und somit den Modellierungsaufwand zu senken wird in [SLT13] vier Modellierungstiefen unterschieden:

- **Idealisierte Funktion**: In dieser Stufe werden die Modelle abstrakt in Form von zeitdiskreten Zustandsautomaten abgebildet, wobei kein Zeitverhalten berücksichtigt wird.
- **Prinzipielle Machbarkeit**: Es wird zusätzlich das Zeitverhalten mit modelliert, somit kann das Modell einfaches physikalisches Verhalten aufweisen.

- **Systemspezifisches Verhalten**: Das Modell dieser Modellierungsstufe wird physikalisch basiert erstellt unter Mitberücksichtigung von Nebeneffekten wie Reibung und Verlustströme. Mit dem Modell in solcher Detaillierung sind Aktivitäten wie Systemverhaltensanalyse und Reglerauslegung möglich.
- **Bauteiloptimierung**: Diese Modellierungsstufe hat den höchsten Detaillierungsgrad. Alle Effekte werden so detailliert wie möglich modelliert. Modell mit dem Abstraktionsgrad ermöglicht somit Auslegung einzelner Komponente.

Bewertung:

Zusätzlich zu den Modellierungstiefen werden hier auch unterschiedliche Modellklassen (Zustandsmodelle, Finite-Elemente-Modelle, Mathematische Modelle usw.) definiert, die auf unterschiedlichen Beschreibungsmitteln beruhen. Jedoch lassen sich Modellklassen nicht eindeutig den Modellierungstiefen zuweisen. Z.B. das Mathematische Modell kann in alle vier Modellierungstiefen eingesetzt werden. Es ist zu sehen, dass die Definition der unterschiedlichen Detailierungsstufen pragmatisch nach Entwicklungsprozess erfolgt. Die Abgrenzung zwischen einzelner Stufe sowie Einsatzgebiete bleiben unklar.

3.5 Simulationsmodellvalidierung

3.5.1 Model Checking

Model Checking oder Modellprüfung ist ein formales Verfahren der analytischen Qualitätssicherung zur Verifikation von Embedded Systemen, Software Engineering und Hardware Design. Es wird dafür konzipiert, nebenläufig arbeitende Programme zuverlässig auf funktionale Richtigkeit, Erreichbarkeit, Sicherheit, Lebendigkeit und Fairness zu prüfen. Dabei wird mit mathematischen Methoden ein Programm auf den Einhalt bestimmter vorher definierter Randbedingungen geprüft und mit einem mathematischen Beweis belegt [Hof13]. Ein etabliertes Werkzeug ist der Simple PROMELA Interpreter- Modellchecker oder kurz SPIN-Modellprüfer.

Nach [Hof13] lässt sich der Verifikationsablauf mit einem Modellprüfer in die drei Phasen Modellierung, Ausführung und Auswertung der Analyse einteilen.

In Schritt eins wird das zu verifizierende Programm in ein Modell überführt. Dies erfolgt bei dem SPIN-Modellprüfer als Beispiel mit der Modellierungssprache Process Meta Language (PROMELA). Das Modell entspricht einem endlichen Zustandsautomaten. Das nachgebildete System verhilft dazu, das gegebene Programm auf die zu prüfende Funktionalität zu minimieren und ermöglicht bei sehr umfangreichen und komplexen

Programmen erst eine effiziente Prüfung. Das Modell muss so beschaffen sein, dass es ausführbar ist.

In Phase zwei wird die Modellchecker-Software das erstellte Modell übergeben und die Verifikation durchgeführt. Der Prüfvorgang verläuft vollautomatisch. Während der Verifikation wird ein Zustandsübergangsgraph des Modells systematisch durchlaufen und ein mathematischer Beweis aufgebaut, ob die geforderten Prüfkriterien eingehalten werden. Im Falle einer Prüfbedingungsverletzung ist ein Gegenbeispiel gefunden und der Verifikationsdurchlauf endet an dieser Stelle.

In der dritten Phase wird das Ergebnis analysiert. Hierbei gibt es drei mögliche Ergebnisse. Zum einen kann das betrachtete Prüfkriterium erfüllt sein, es kann nicht erfüllt sein oder die Modellgröße wächst über den verfügbaren Arbeitsspeicher hinaus und die Prüfung wird abgebrochen. Sobald ein Prüfkriterium erfüllt ist, wird das nächste verifiziert und falls alle erfüllt sind, so entspricht das Modell den Anforderungen. Sobald ein Gegenbeispiel eines Prüfkriteriums gefunden wird, entspricht das Modell nicht den Forderungen. Bei umfangreichen und komplexen Problemstellungen kommt es dabei häufig zur Zustandsexplosion, da der Zustandsmaschine sehr viele Zustände beinhaltet und den zur Verfügung stehenden Arbeitsspeicher überschreitet. Der Zustandsraum nimmt exponentiell mit der Problemgröße zu. Durch Optimierungstechnologien wie symbolische Modellprüfung, Abstraktion und Partial-Order-Reduction wird diese Situation abgewendet [BK08a]. Bei Auftreten eines Gegenbeispiels wird das Zustandekommen als ein chronologischer trace-Bericht ausgegeben, der den exakten Verlauf der Modellausführung bis zum Erreichen des Gegenbeispiels aufzeigt. Dieser Bericht ist für die Fehleranalyse elementar und ermöglicht eine gezielte Fehlerbehebung.

Bewertung:

Die Stärke der Modellprüfung zeigt sich durch seine mathematische Theorie (graph algorithmus, data structures and logic), welche einen Beweis der geforderten Verifikationskriterien liefert. Es kann grundsätzlich für Teilbereiche wie die Verriegelungsprüfung eingesetzt werden und die Zuverlässigkeit der Programme somit erheblich erhöhen. Da jedes Kriterium explizit angegeben werden muss, eignet sich ein solches Verfahren nicht als kompletter Ersatz für die simulativen Verfahren im Engineeringprozess. Beispielsweise können Performance und zeitliche Aspekte nicht untersucht werden.

3.5.2 Modellbasierter Test

Heutzutage werden die meisten Testfälle manuelle durch menschliche Testentwickler erstellt und ausgeführt, was zu einer unzureichenden Testabdeckung führt [AS09]. Nach [RBGW12]

betragen die Testvorbereitung sowie die nachgelagerten Arbeiten (z.B. Auswertung, Berichterstellung, Archivierung etc.) nur 15% der gesamten Aufwände im klassischen Testverfahren. Die anderen 85% fallen an die Erstellung der Testfälle sowie Testdurchführung. Diese beiden Aspekte können durch den modellbasierten Test (MBT) den größten Nutzen generieren. In [RBGW12] wird für MBT eine pragmatische Definition gegeben: „Modellbasiertes Testen umfasst mindestens einen der beiden folgenden Aspekte:

- die Nutzung von Modellen für die Automatisierung von Testaktivitäten sowie
- die Modellierung von Artefakten im Testprozess.“

In [MKD15] [MRKW16] [SMT+16] wird dieser Prozess in Modellierung, Testfallgenerierung und Testdurchführung geteilt.

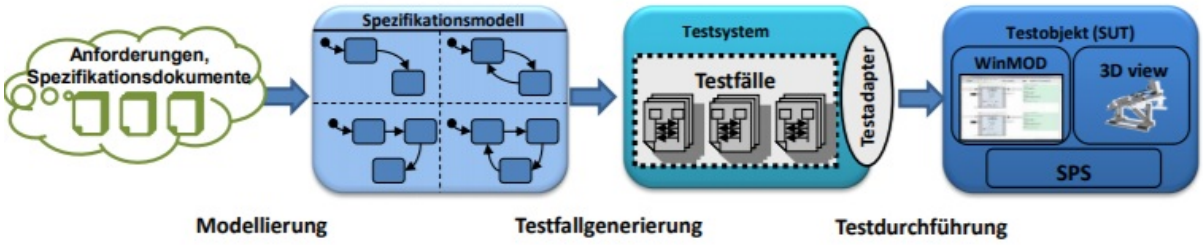


Abbildung 3.4: Testaktivitäten im modellbasierten Testprozess [MRKW16]

Die automatische Generierung der Testfälle setzt ein Spezifikationsmodell voraus, das unter Verwendung geeigneter Methoden und Werkzeuge die Testanforderung spezifiziert. Wenn die Anforderung in einer formalisierten Form beschrieben wird, lassen sich Testfälle automatisiert aus dem Modell generieren. Für die Formalisierung von Anforderungen werden in [MRKW16] als Beschreibungsmittel Sequenzdiagramme gewählt. Für die textuelle Darstellung der Sequenzdiagramme hat der Autor eine textbasierte Sprache namens „avanti RDL“ sowie dazugehöriges Tool entwickelt. Mit dem prototypischen Werkzeug lässt sich die Anforderung in textueller Form eingeben und wieder grafisch als Sequenzdiagramme anzeigen.

Auf dieser Basis kann mit der Modellsynthese das Spezifikationsmodell automatisch erstellt werden (vgl. [WS00] [WS02]). Ergebnis der Modellsynthese ist ein Zustandsautomat, der das Verhalten des SUT (system under test) aus Testersicht beschreibt und als Ausgangsmodell für die Testgenerierung dient.

Eine Möglichkeit zur modellbasierten Testfallgenerierung wird in [MKD12] und [Kra11] vorgeschlagen. Dieser Ansatz basiert auf der Erweiterung des Petrinetzes zu SPENAT (Sichere Petrinetze mit Attributen). Der spezifizierte Zustandsautomat wird in die SPENAT überführt. Aus der Entfaltung lassen sich abstrakte Testfälle (Testsequenz) generieren.

Bei der Testdurchführung werden die generierten abstrakten Testsequenzen in einer Testumgebung als konkrete Testfälle ausgeführt, die aus Teststimuli und -reaktion bestehen. Die Testumgebung beinhaltet ein Werkzeug und Testinfrastruktur, die eine Anbindung mit dem SUT erstellt. Das Testwerkzeug kann sowohl als Eigenentwicklung für spezielle Anwendungsfälle als auch Standardwerkzeuge wie TTCN-3 [GHR+03] sein.

Bewertung:

Modellbasierter Test nach [MKD15] [MRKW16] bietet eine komplette Modell- und Werkzeugkette von Anforderungsspezifikation bis zur Testdurchführung. Jedoch lässt sich der Abdeckungsumfang der abgeleiteten Testfälle nicht bewerten, da die Qualität des Spezifikationsmodells von den eingegebenen Anforderungen, die die Testziele bestimmen, abhängig ist.

3.6 Handlungsbedarf

In Kapitel 3 wird der aktuelle Forschungs- und Entwicklungsstand der virtuellen Inbetriebnahme, insbesondere Entwicklung der Simulationsmodelle einer Komponente, analysiert und gegenübergestellt. Jedoch sind noch Defizite in der bisherigen Forschung und Entwicklung vorhanden, die eine noch weiter tiefgreifende Anwendung der virtuellen Inbetriebnahme behindern. Keine Ansätze in Kapitel 3 sind in der Lage, für solche vorhandene Defizite ein gutes Lösungskonzept zu bieten.

> **Defizit 1** :
> *Kein geeignetes Vorgehensmodell für den Entwicklungsprozess in der virtuellen Inbetriebnahme ist vorhanden.*

Heutzutage folgen viele Entwicklungsprozesse der Simulationsmodelle für die virtuelle Inbetriebnahme eine lineare Reihenfolge, die sich stark an den klassischen Engineeringsprozess anlehnen. Es gibt auch Ansätze, die auf einen baukastenbasierten Prozess beruht, jedoch fehlt es eine systematische Vorgehensweise. (vgl. Abschnitt 3.2 und Abschnitt 3.3.2)

> **Defizit 2** :
> *Fehlerverhalten sind nicht mit modelliert.*

Die betrachteten Modellierungsansätze basieren auf vorhandene Simulationstechnologien, die unterschiedliche Systemverhalten nachbilden (vgl. Abschnitt 3.1). Im Sinne der virtuellen Inbetriebnahme wird jedoch angefordert, zusätzlich zum geplanten Ablauf (Gutverhalten), Fehlerbehandlungen in den Steuerungsprogrammen auch getestet werden sollen. Dies stellt heraus, dass Fehlerverhalten auch in einem Simulationsmodell

mit modelliert werden. Es sind keine Ansätze bekannt, dieses Thema systematisch zu behandeln.

Defizit 3 :
Es fehlt für verschiedene Detaillierungsgrade ein skalierbarer Modellierungsansatz für unterschiedliche Anwendungsfälle.

Generell gilt ein Grundsatz für die Erstellung eines Simulationsmodells: so einfach wie möglich und so detailliert wie nötig. D.h. ein Simulationsmodell soll für sein Anwendungszweck zugeschnitten sein. Zwar bestehen bereits viele Ansätze, die die Modellierungstiefe nach unterschiedlichen Aspekten definieren (vgl. Abschnitt 3.4). Jedoch ist kein Ansatz davon in der Lage, ausgehend von den Anwendungsfällen in der virtuellen Inbetriebnahme in der Fertigungstechnik allgemein, die Detaillierungsstufe für Komponentenmodell zu definieren.

Defizit 4 :
Testdurchführung in der virtuellen Inbetriebnahme erfolgt meistens noch manuell.

Heutzutage werden die einzelnen Testfälle noch manuell ausgeführt. Dies erfordert hohen Zeit- und Personalaufwand und erschwert den effizienten Einsatz der virtuellen Inbetriebnahme. Zudem müssen für die Testdurchführung für die Störverhalten manuell das Simulationsmodell eingreifen, was nicht nur zusätzliche interne Kenntnisse über die Modelle benötigt, sondern auch eine Fehlermöglichkeit darstellt, wenn das Modell nach der Testdurchführung nicht richtig zurückgesetzt wird.

3.7 Lösungskonzepte

Um die im letzten Abschnitt identifizierte Defizite lösen zu können, werden im Folgenden Lösungskonzepte erläutert, die in Rahmen dieser Arbeit im Detail zu behandeln sind.

Es soll zunächst ein Vorgehensmodell gefunden werden, das auf dem klassischen V-Modell beruht und so modifiziert wird, dass die Vorgehensweise für die virtuelle Inbetriebnahme dadurch besser spezifiziert werden kann (-> Kapitel 4). Ziel ist es, ein neues Geschäftsmodell für virtuelle Inbetriebnahme auf dieser Basis abzuleiten, nach dem die Modellentwicklung für virtuelle Inbetriebnahme in Komponenten- und Anlagenebene zu trennen ist. Ergebnisse der Entwicklung auf Komponentenebene sind validierte Komponentenmodelle, die in eine kuratierte Bibliothek aufgenommen werden können und die Basis für die Modellentwicklung auf Anlagenebene bilden. Modellbildung auf Komponentenebene stellt im weiteren Teil dieser Arbeit eben den Schwerpunkt dar.

Um das Defizit 2 und Defizit 3 in Abschnitt 3.6 zu behandeln wird die Verhaltensmodellierung einer Komponente im Anschluss näher betrachtet (-> Kapitel 5). Für diverse Anwendungsszenarien sollen unterschiedliche Modellierungsansätze vorgeschlagen werden, die die Modelle jeweils mit passenden Modellierungstiefen skalierbar abbilden können. Der Testumfang in der virtuellen Inbetriebnahme beschränkt sich nicht nur auf das fehlerfreie Verhalten. Häufig wird auch mit getestet, ob die Steuerung auf bestimmte Störungen reagieren bzw. richtige Störmeldungen erzeugen kann. Es soll auch sichergestellt werden, dass die Störszenarien möglich vollständig identifiziert werden können.

Das in Kapitel 4 definierte Vorgehensmodell schlägt einen komponentenbasierten Entwicklungsprozess vor. Ein Austauschformat soll eingeführt werden, welches den Austausch der Komponentenmodelle ermöglicht (-> Kapitel 6). Das Austauschformat soll herstellerneutral sein und somit unabhängig von den Entwicklungsplattformen, die der Modellentwickler bzw. ein Unternehmen verwenden.

Am Ende soll ein Testframework entwickelt werden, das die Testdurchführung in der virtuellen Inbetriebnahme automatisiert unterstützt (-> Kapitel 7). Es ist derzeit kein Werkzeug bekannt, das eine automatische Testdurchführung ermöglicht, die den Modellierungsaufwand in der virtuellen Inbetriebnahme deutlich verringern kann. Dies soll sich nicht nur auf den Test des normalen Verhaltens beschränken. Testfälle, die zum Test des Fehlerverhaltens entwickelt werden, sollen auch in das Modell aufgenommen und deshalb auch im Werkzeug unterstützt werden.

Die Lösungskonzepte zur Behandlung der beschriebenen Probleme sind in Abbildung 3.5 in einer Übersicht dargestellt.

Abbildung 3.5: Lösungskonzepte dieser Arbeit

4 Vorgehensmodell für die Modellentwicklung

4.1 Einleitung

In der Praxis wird ein mechatronisches Anlagenmodell mit der Steuerungseinheit über eine Kommunikationsschnittstelle gekoppelt. Diese Kombination bildet die Plattform zum Test der Steuerungssoftware und zur Erkennung von Fehlern der Anlagenfunktion beim Zusammenspiel der Komponenten in der Anlage. Dabei kommt es jedoch häufig zu folgenden unerwünschten Situationen:

- Aufgetretene Fehler können sowohl durch Fehler in den virtuellen Komponenten als auch in der zu testenden Steuerungssoftware liegen. Diese Situation erschwert die Fehlerbehebung erheblich.
- Wird bei der Steuerungssoftwareentwicklung nur mit den Informationen der fertig entworfenen Anlage gearbeitet, so können sich Fehler weiter ausbreiten. Dieselben Fehler können gleichzeitig in der virtuellen Anlage und in der Steuerungssoftware aufgenommen werden, die möglicherweise sich kompensieren und damit nach außen nicht zu erkennen sind.

Ziel dieses Kapitels ist, die aktuelle Vorgehensweise bei der Entwicklung von mechatronischem Anlagenmodell zu verbessern. Dafür wird zunächst eine Einführung zu Vorgehensmodellen gegeben, darauf aufbauend wird ein neues Vorgehensmodell vorgestellt, das die oben geschilderte Problematik behandelt. Inhalt dieses Kapitels basiert teilweise auf der Vorveröffentlichung des Autors in [LD12].

4.2 Etablierte Vorgehensmodelle

Die zunehmende Komplexität bei der Entwicklung eines Produktionssystems erfordert bei der Umsetzung eine logische Ordnung der auftretenden Aufgaben und Aktivitäten. Das primäre Ziel eines Anlagenentwicklungsprozesses ist das Schaffen einer neuen oder

die Verbesserung einer bestehenden Anlage. Um dieses Ziel erreichen zu können, bedarf es eines strukturierten Vorgehens, das den Ablauf des Problemlösungsprozesses genau beschreibt. Es soll der Ablauf der Entstehung der Komponente und der Einbindung in die verschiedenen Entstehungsphasen festgelegt werden. Dadurch wird eine strukturierte und organisierte Arbeitsweise definiert und damit die Qualität des Produktes (hier die Anlage) sichergestellt. Diese Anforderung wird durch ein Vorgehensmodell erfüllt, das die Zeitspanne beschreibt, die mit der Konzipierung eines Produkts beginnt und mit der Fertigstellung oder Außerbetriebnahme endet [IEE90]. Darüber hinaus sind in vielen Branchen schon etablierte Normen und Richtlinien zur Qualitätssicherung während des Produktentstehungsprozesses vorhanden.

Im Folgenden werden einige Vorgehensmodelle, die entweder allgemeingültig oder branchenüblich standardisiert sind, vorgestellt.

4.2.1 Wasserfallmodell

Eines der am weitesten verbreiteten Vorgehensmodelle ist das Wasserfallmodell, das von W. W. Royce in [Roy70] vorgeschlagen wird. Das Wasserfallmodell zeichnet sich durch die sequentielle Struktur aus. Der ganze Prozess wird nach unterschiedlichen Phasen gegliedert. Die Anzahl der Phasen und deren Benennungen unterscheiden sich in unterschiedlichen Literaturquellen [Roy70] [BEP+82] [Jen01]. Dabei muss eine Phase vor Beginn der nächsten Phase vollständig abgeschlossen werden.

Nach [Roy70] kann eine abgeschlossene Phase nicht wiederholt werden. Dies hat den Nachteil, dass spätere Änderungen in System nicht mehr berücksichtigt werden. In [BEP+82] wird eine Rücksprungstruktur hinzugefügt, bei der die Ergebnisse bei Abschluss jeder Phase überprüft werden können (vgl. Abbildung 4.1). Somit können später entdeckte Fehler iterativ zurückverfolgt und an der richtigen Phase korrigiert werden.

Der Vorteil dieses Modells besteht in der Einfachheit, der übersichtlichen und verständlichen Struktur und damit verbundenen wenigen administrativen Aufwand. Allerdings führen das starre Phasenmodell und die strikte Phasentrennung zu langen Entwurfszyklen. Folglich bietet sich das Wasserfallmodell für kleine Projekte an, die eine kurze Projektlaufzeit und einfache Projektstruktur haben.

4.2.2 Spiralmodell

Um den Nachteilen von Wasserfallmodell entgegenzukommen, wird das Spiralmodell von Boehm eingeführt [Boe88]. Das Spiralmodell stellt ein iterativ-inkrementelles Modell

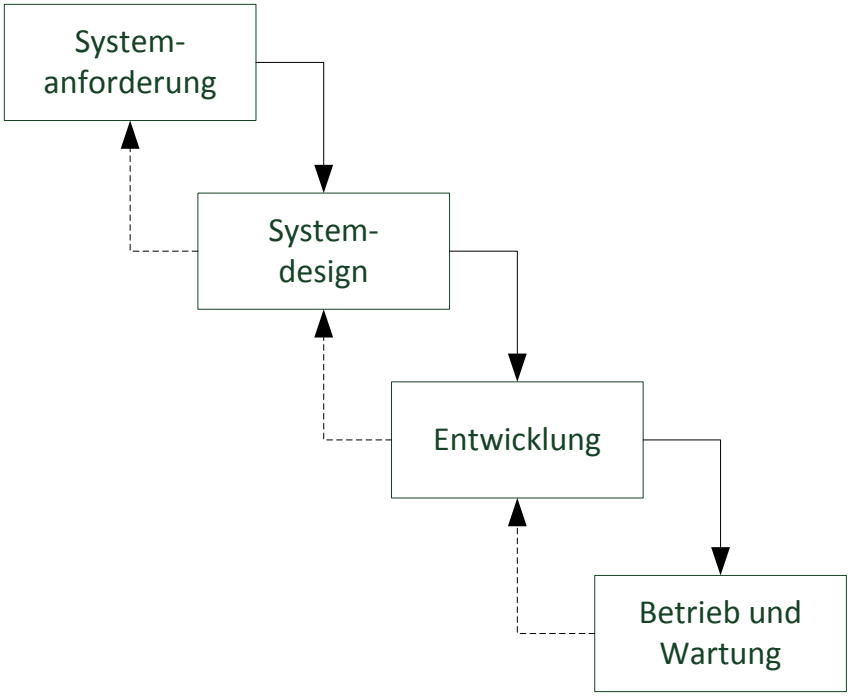


Abbildung 4.1: Wasserfallmodell mit Rücksprung (in Anlehnung an [BEP+82])

dar, welches vier fest definierte Schritte mehrmals zyklisch durchlaufen, bis ein gewisser Reifegrad erreicht ist (vgl. Abbildung 4.2):

- Ziel und alternativen Generation
- Risikoanalyse, Entwurf und Erstellung eines Prototyps
- Entwicklung und Anwendertest
- Auswertung der Iteration und Planung der nächsten Iteration bzw. Stopp des Projekts

Da jeder Umlauf dem Durchlaufen eines Entwicklungszyklus des Wasserfallmodells entspricht, nähert sich schrittweise der Prototyp dem fertigen Produkt mit der mehrstufigen Iteration, wobei die Qualität sich verbessert nach jedem Zyklus. Fehler können früh erkannt und rechtzeitig korrigiert werden.

Zusätzlich wird eine intensive Risikoanalyse in den Prozess eingeführt, die die Ziele der nächsten Iteration bestimmt. Die Anwendung des Spiralmodells setzt ein hohes Know-how des Projektmanagers über die Handhabung von Risiken voraus.

In der Praxis ist das Spiralmodell vor allem für große Projekte geeignet aufgrund der großen Komplexität bei der Durchführung sowie dem hohen Organisationsaufwand.

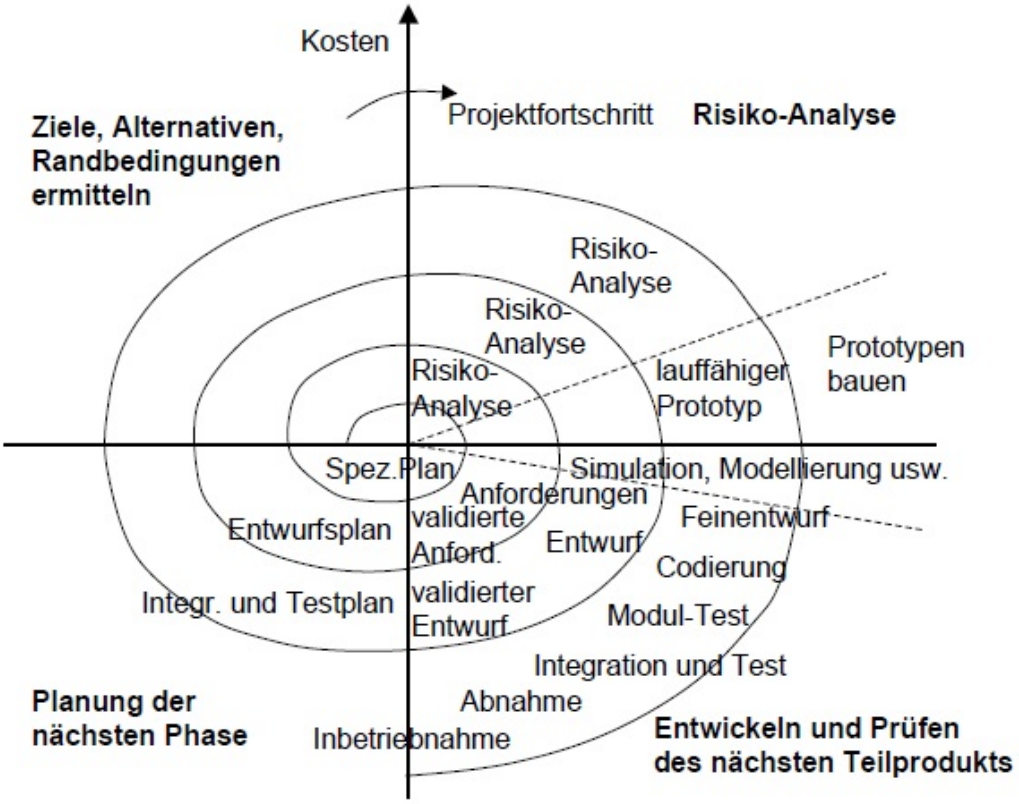


Abbildung 4.2: Spiralmodell nach [Boe88]

4.2.3 V-Modell

Das V-Modell erweitert das Wasserfallmodell um die Qualitätssicherungsmaßnahmen, stellt den einzelnen Spezifikations- und Entwicklungsschritten korrespondierende Testschritte gegenüber. Der Namen des V-Modells erklärt sich aus der V-förmigen Darstellung, die die beiden Aspekte veranschaulichen (Abbildung 4.3).

Der Prozess wird in einen linken Ast für die Entwicklungsphase und einen rechten Ast für die Verifizierung und Validierung aufgeteilt.

Die Spezifikation und Systementwicklung erfolgen einen Top-Down-Ansatz. Nach der Anforderungsanalyse wird die Entwicklung sukzessiv von Systemebene auf Teilsystem bis hin zu Komponenten aufgegliedert. Jede Entwicklungsstufe auf der linken Seit wird durch einen entsprechenden Verifikations- und Validierungsschritt auf der rechten Seite getestet. Die Testausführung wird in umgekehrter Reihenfolge wie die Entwicklungsphasen durchgeführt und erfolgt somit den Bottom-Up-Ansatz. Bei der Verifikation wird die Korrektheit der Entwicklung überprüft. Zuerst wird fertige Komponente getestet. Jede verifizierte Komponente wird in der nächsten Stufe mit anderen Komponenten zu einer nächstgrößeren Funktionseinheit integriert. Dieser Vorgang wiederholt sich, bis das ganze System zusammengebaut. Anschließend werden die Anforderungen der Systemspezifikation

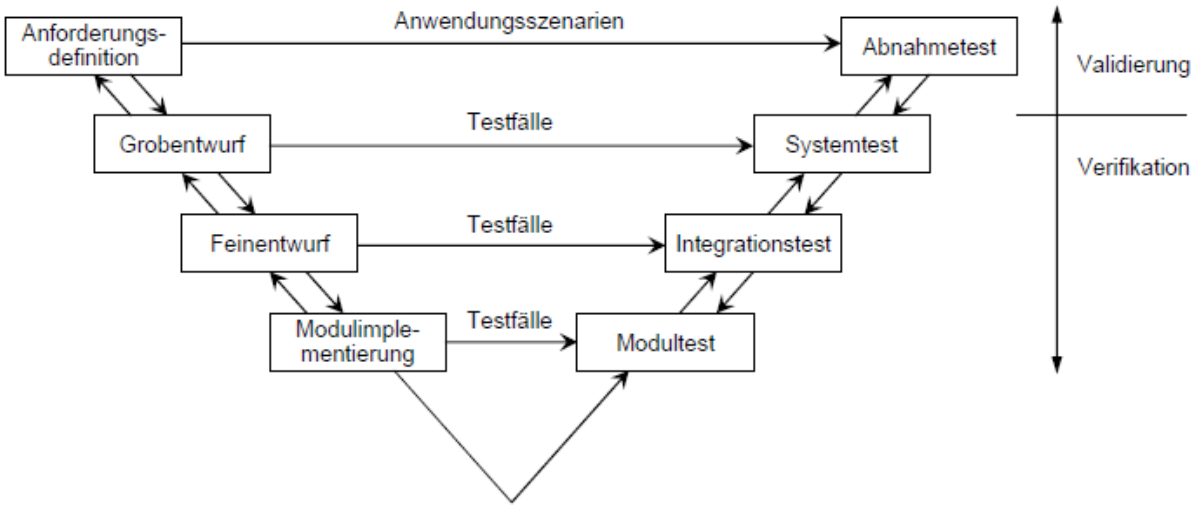


Abbildung 4.3: V-Modell nach [AAB+14]

getestet. Der Abnahmetest wird hier mit der Validierung gleichgestellt, bei der die Eignung eines Produktes bezogen auf Abnahmekriterien überprüft wird.

Durch die starke Trennung des Entwicklungs- und Testprozesses im V-Modell ist es möglich, Testfälle nach jeder Entwurfsphase abzuleiten. Es werden überprüft, ob gegebene Anforderung an das System erfüllt wurden. Damit wird die Qualität der jeweiligen Module frühzeitiger aufgedeckt im Vergleich zu Vorgehensmodellen wie Wasserfall- oder Spiralmodell. Des Weiteren ist die parallele Entwicklung von Subsystemen nach dem Entwurf denkbar. Somit wird die Entwicklungszeit verkürzt.

4.2.4 Das Vorgehensmodell nach ISO 26262

Die ISO 26262 („Road vehicles - Functional safety“) [ISO11] ist ein internationaler Standard zur Erreichung und Umsetzung von funktionaler Sicherheit in der Automobilindustrie. Dieser Standard umfasst insgesamt 1269 zu berücksichtigenden Vorgaben für die Entwicklung und den Test von softwarebasierten Fahrzeugsystemen mit hohen Sicherheitsansprüchen [Hal13].

ISO 26262 definiert ein Prozessrahmenwerk und Vorgehensmodell zusammen mit geforderten Aktivitäten und Arbeitsprodukten (Dokumentation, Projektplanung, etc.) sowie anzuwendenden Methoden, die während der Entwicklung, Produktion, Betrieb, Wartung und Entsorgung von sicherheitsbezogenen Systemen von Kraftfahrzeugen durchgeführt werden müssen [Hil12].

Insgesamt besteht ISO 26262 aus zehn Bänden (Abbildung 4.4). In Band 3 wird potentielle Gefährdungen identifiziert und in unterschiedlichen Stufen klassifiziert. Sicherheitsziele

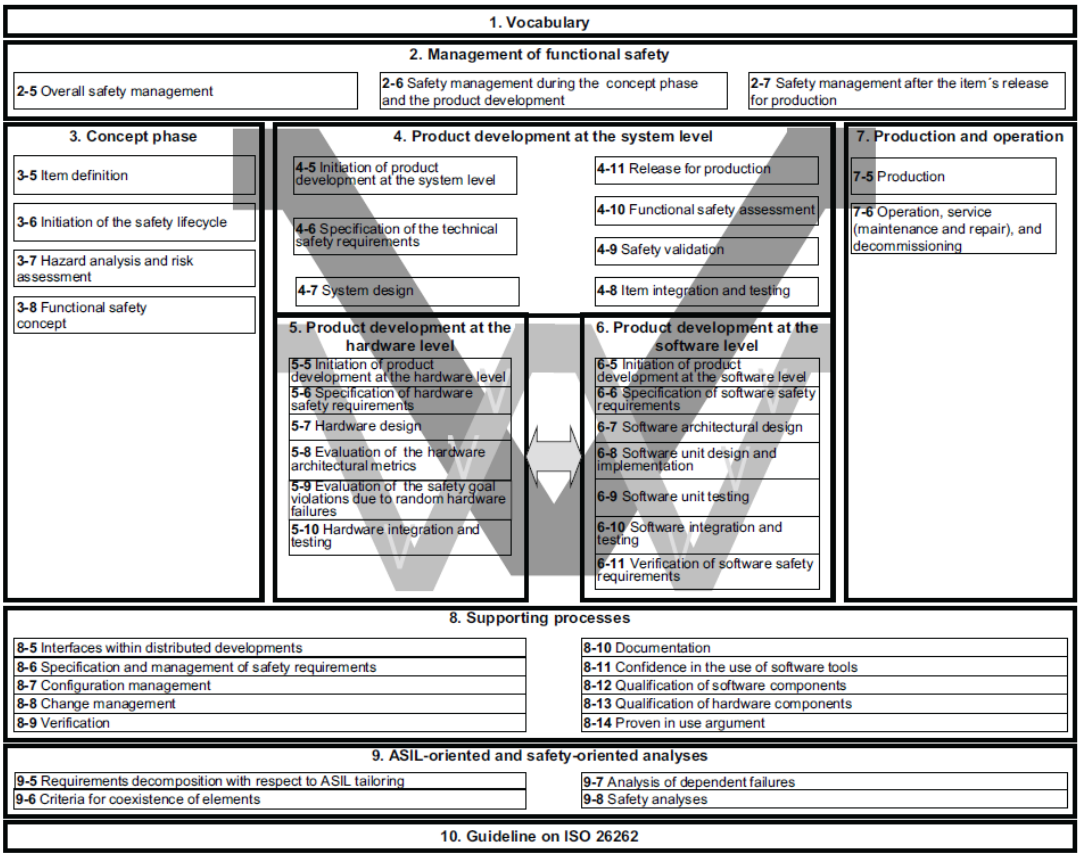


Abbildung 4.4: Überblick über die ISO 26262 (englisch) [ISO11]

werden abgeleitet. Dementsprechend werden Sicherheitsanforderungen spezifiziert. Band 7 befasst sich mit dem Vorgehen beim Erstellen eines Produktions- und Installationsplans sowie die Anforderung an die Herstellung, Verwendung, Wartung und Stilllegung für sicherheitsrelevante Systeme.

Band 3 bis Band 7 beschreiben den Lebenszyklus von Sicherheitsfunktionen und bilden den Kern des Standards. Der betrachtete Lebenszyklus besteht hauptsächlichen aus drei Phasen:

- Konzeptphase (Band 3)
- Produktentwicklung (Band 4, 5 und 6)
- Produktion und Operation (Band 7).

Auf den Aspekt des Lebenszyklus wird in der Dissertation nicht weiter betrachtet.

In der Produktentwicklungsphase wird die Vorgehensweise zur Entwicklung bzw. Test des Produkts definiert. Band 4 der ISO 26262 beschäftigt sich mit der Ableitung von Sicherheitsanforderungen sowie der Verfeinerung der Systemarchitektur. In Band 5 und 6 wird die Systementwicklung auf Hardware- und Softwarekomponenten unterteilt. Der jeweilige Entwicklungsprozess der Hardware- bzw. Softwareteile wird betrachtet.

Hier orientiert sich die Darstellung an drei V-Modellen mit geschachtelter Struktur. Während das übergeordnete V-Modell die Produktentwicklung beschreibt (Band 4), sind die Teilprozesse für die Hardware- und Softwareentwicklung jeweils in eigenen V-Modellen definiert (Band 5 und 6). Durch diese Struktur sind die Hardware- und Softwareentwicklung eng in die Systementwicklung integriert.

4.2.5 Das Vorgehensmodell nach GAMP

Im Bereich der Pharmaindustrie spielt die Qualitätssicherung der Produkte eine zentrale Rolle, da hier Qualitätsabweichungen direkte Auswirkungen auf die Gesundheit der Verbraucher haben können. Die pharmazeutische Herstellung wird deshalb durch strenge Gesetzgebung und Richtlinien reguliert. Unter GMP (Good Manufacturing Practice) versteht man den Oberbegriff international anerkannter Richtlinien zur Qualitätssicherung in der pharmazeutischen Produktion. GMP wird in verschiedenen Ländern in Form von Gesetzen und Regulierung festgelegt, z.B. der US Code of Federal Regulations, Titel 21, Part 210 (21 CFR Part 210) [FA12], der EU-GMP-Leitfaden [ECG], der Q7A von ICH (International Conference on Harmonization) [HI99], unter denen sind die Konzepte in einem höheren Niveau ähnlich. Die Unterschiede liegen hauptsächlich in den detaillierten Erklärungen. Die ISPE (International Society for Pharmaceutical Engineering) kam der

wachsenden Bedeutung der Automatisierungssysteme 1995 mit dem Leitfaden Good Automated Manufacturing Practice (GAMP) nach, der aktuell in der 5. Version (GAMP 5) vorliegt [GAM08].

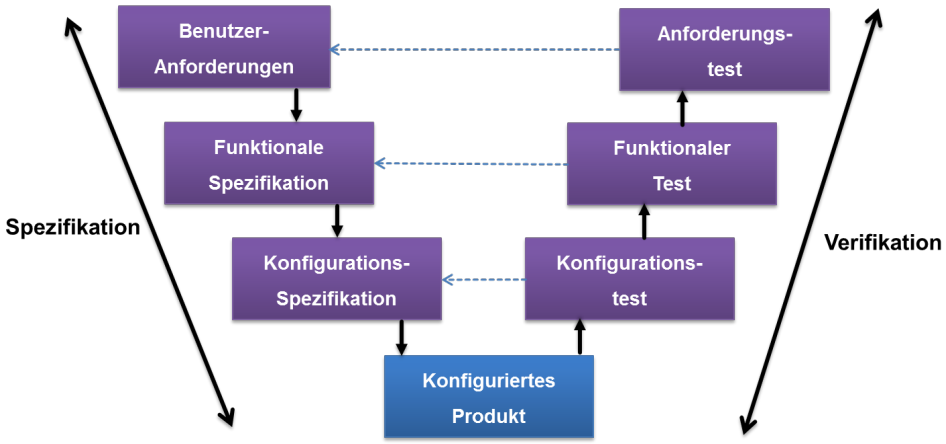


Abbildung 4.5: V-Modell für konfigurierte Anlagen nach GAMP 5 [GAM08]

Der GAMP-Leitfaden greift die wichtigsten Prinzipien und Richtlinien auf und ermöglicht den Lieferanten von automatisierten Systemen von den verbindlichen Vorschriften der Inspektionsbehörde zu qualifizieren. Dabei werden die entsprechenden Gesetze aus Europa und Nordamerika berücksichtigt. Mittlerweile gilt GAMP 5 als der weltweit etablierte Standard für die Validierung automatisierter Systeme in der Pharmaindustrie. GAMP 5 greift den Lebenszyklusansatz des Anlagenengineerings auf und bereichert ihn um das Validierungskonzept. Die Kernidee liegt darin, dass sich die Anlagenqualifizierung nicht als eine einzelne Phase in den oben genannten Lebenszyklus betrachten lässt, sondern dass sie begleitend in den ganzen Lebenszyklus integriert werden muss.

Diese Vorgehensweise wird in GAMP 5 als V-Modell für konfigurierte Anlagen anschaulich dargestellt (Abbildung 4.5). Diese V-Modellvariante setzt im Wesentlichen auf dem bereits bekannten V-Modell auf (vgl. Abschnitt 4.2.3) mit der zusätzlichen Berücksichtigung, dass die Module nicht neu zu entwickeln sind, sondern aus vorhandenen Bibliotheken (Module werden in den Bibliotheken oft auch als Bausteine benannt) entnommen und auf den Anwendungsfall konfiguriert werden. Nach dem V-Modell für konfigurierte Anlagen wird jedem Spezifikationsschritt auf dem linken Ast die entsprechende Qualifizierungsmaßnahme auf dem rechten Ast zugeordnet. Die Qualifizierungsmaßnahmen dienen der Verifikation der Spezifikation.

Im folgenden Kapitel soll auf die Problematik des Entwicklungsvorgehens des mechatronischen Anlagenmodells in der virtuellen Inbetriebnahme eingegangen und in Anlehnung an die oben vorgestellten Vorgehensmodelle das V-Modell erweitert werden. Das Ziel ist, aus dem neuen Vorgehensmodell einen Entwicklungsprozess für das mechatronische Anlagenmodell zu finden, der eine komponentenbasierte Modellierung ermöglicht.

4.3 Vorschlag eines modifizierten Vorgehensmodells für die Entwicklung eines mechatronischen Anlagenmodells

4.3.1 Problematik

Voraussetzung für einen gleitenden Einsatz der virtuellen Inbetriebnahme ist, dass die Funktionalität der virtuellen Anlagen den realen Anlagen entspricht. Um diese Anforderung zu erfüllen, müssen die virtuellen Anlagen einen Qualitätssicherungsprozess durchlaufen, der in äquivalenter Weise durchzuführen ist, wie dies bei realen Anlagen üblich ist.

Derzeit wird das mechatronische Anlagenmodell jedoch nach einem linearen Vorgehen aufgebaut (siehe auch 4.2.1). Beim Aufbau von virtuellen Maschinen und Anlagen muss für jede Komponente eine simulationsfähige virtuelle Komponente mit allen für die Simulation benötigten Aspekten (z.B. Geometrie, Kinematik, Verhalten, Schnittstellen) zur Verfügung stehen [KBR10]. Nach der Erstellung der kinematisierten Anlage und der Prozessbeschreibung der Abläufe dieser Anlage, wird aus diesen Informationen das Steuerungsprogramm erstellt. Im virtuellen Anlagenmodell wird das Verhalten der Betriebsmittel ergänzt und mit dem kinematischen Modell verbunden.

Die Idee bei diesem Vorgehen ist, entlang der Entwicklungsschritte jeweils so viele Daten wie möglich weiter zu verwenden. Bei der dann folgenden virtuellen Inbetriebnahme kommen alle virtuellen und steuerungstechnischen Komponenten erstmalig zusammen. Das ist viel zu spät. Denn es ist zu beachten, dass beim oben beschriebenen Wasserfallmodell Validierungsschritte für die einzelnen Komponenten und Schritte fehlen. Dies hat zur Folge, dass unvollständige bzw. fehlerhafte Ergebnisse direkt von einer vorherigen Phase zu einer Folgephase fortgeführt werden können. Dadurch werden Fehler erst sehr spät erkannt und aufwendige und kostspielige Rückkopplungszyklen aus einer späten Phase eines Projektes sind unvermeidbar.

Es wird zunächst der Aufbauprozess einer realen Anlage betrachtet. Jede reale Komponente ist in der Regel ein fertiges Produkt, das beim Hersteller einen Qualitätssicherungsprozess von ihrer Entwicklung, über die Herstellung bis zur Lieferung entsprechend dem branchenüblichen Vorgehensmodell durchlaufen hat. Beim Zusammenbau auf der Baustelle kann folglich von den funktionstüchtigen Komponenten ausgegangen werden [VDI04]. Fehler sind deshalb hauptsächlich durch deren Kopplung und Zusammenwirken zu erwarten.

Diese Vorgehensweise zur Qualitätssicherung hat, wie im Folgenden vorgeschlagen, für die gesamte Entwicklung aller virtuellen Komponenten zu erfolgen, d.h. eine virtuelle Komponente ist ein qualitätsgesichertes Produkt bzw. ist ein additiver Bestandteil der

realen Komponente. Die virtuellen Komponenten haben ebenfalls einen Qualitätsmanagementprozess durchzulaufen. Wie bei der Installation der realen Anlage werden diese qualitätsgeprüften virtuellen Komponenten unter Beachtung von qualitätssichernden Maßnahmen zu virtuellen Anlagen zusammengesetzt. Gemeinschaftlich bilden die Entwicklung virtueller Komponenten und die Erstellung der Anlage den Validierungsprozess für virtuelle Anlagen.

4.3.2 Modifikation des V-Modells

Deswegen wird bei dem neuen Validierungskonzept ein neues Vorgehen vorgeschlagen. Die Grundidee ist eine funktionale Diversität. Dies basiert auf einer funktionalen Trennung in anlagentechnische und steuerungstechnische Aspekte. Es existiert für jede Komponente je Teilaspekt (Geometrie, Komponentenverhalten usw.) ein Virtualisierungsbaustein (d.h. Dateien mit diesem Aspekt beschreibenden Modellen) und ein Steuerungsbaustein. Die Bausteine werden separat entlang des Qualitätsmanagementprozesses entwickelt und separat in ihren Entwicklungsumgebungen getestet. Dadurch wird eine eigenständige Qualitätssicherung der einzeln entwickelten Bausteine erzielt. Neu entwickelte Bausteine werden in die projektübergreifende Bibliothek eingesteckt, damit diese in neuen Projekten verwendet werden können. In den Bibliotheken sollten die einzelnen Komponenten kategorisiert (z.B. nach funktionalem Aspekt) werden, damit sie leicht auffindbar sind.

Aus Abschnitt 4.2 ist zu erkennen, dass das V-Modell, aufgrund der zusätzlichen Integration der Qualitätssicherung, sich gut eignet für den Entwicklungsprozess des Simulationsmodells in der virtuellen Inbetriebnahme. So hat die Entwicklung einer virtuellen Komponente auch einen V-Modell-Prozess durchzulaufen, der eine Anforderungs-, Design- und Testphase besitzt. Dies entspricht auch dem Entwicklungsprozess eines mechatronischen Anlagenmodells.

Da ein Anlagenmodell aus zahlreichen virtuellen Komponenten besteht, reicht die Darstellung der Entwicklung durch ein einfaches V-Modell nicht aus. Der Entwicklungsprozess von sowohl einer virtuellen Komponente als auch einem mechatronischen Anlagenmodell muss durch ein eigenständiges V-Modell definiert werden. Inspiriert von dem Vorgehensmodell nach ISO 26262 (vgl. Abschnitt 4.2.4), indem die Teilprozesse der Software- und Hardwareentwicklung jeweils in eigenen V-Modellen definiert werden, wird im Folgenden eine kaskadierte V-Modell-Struktur eingeführt.

4.3.3 Das kaskadierte V-Modell

Die Entwicklung der einzelnen Komponenten und deren Integration in die gesamte Anlage erfolgen in Prozessen basierend auf den zwei getrennten V-Modellen-Prozessen.

4.3.3.1 Das V-Modell für den Entwicklungsprozess der virtuellen Komponenten

Dieses V-Modell hat das Ziel, eine eigene Qualitätssicherung für die einzelnen zu entwickelnden Bausteine zu garantieren, d.h. diese Virtualisierungs- und Steuerungsbausteine sollen vor der Integration in die Bibliothek als funktional richtig nachgewiesen werden. Dadurch ist es möglich, die Fehler der einzelnen Bausteine schon vor deren Nutzung im virtuellen Anlagenmodell zu erkennen und zu korrigieren sind. Die Komponenten basieren auf den Anforderungen, die sich aus dem Anlagenentwurf ableiten (Pflichtenheft). Das verwendete Komponenten-V-Modell nutzt direkt diese Anforderungen als Input. Für jede Komponente zergliedert man die virtuelle Komponente zunächst in ihre funktionalen Aspekte, z.B. die Kinematik und das Verhalten. Danach erfolgt die Entwicklung der entsprechenden Bausteine. Sind alle Bausteine einer Komponente entwickelt, soll die Integration der einzelnen Bausteine zu einer virtuellen Komponente erfolgen, die wieder eigenständig zu testen ist.

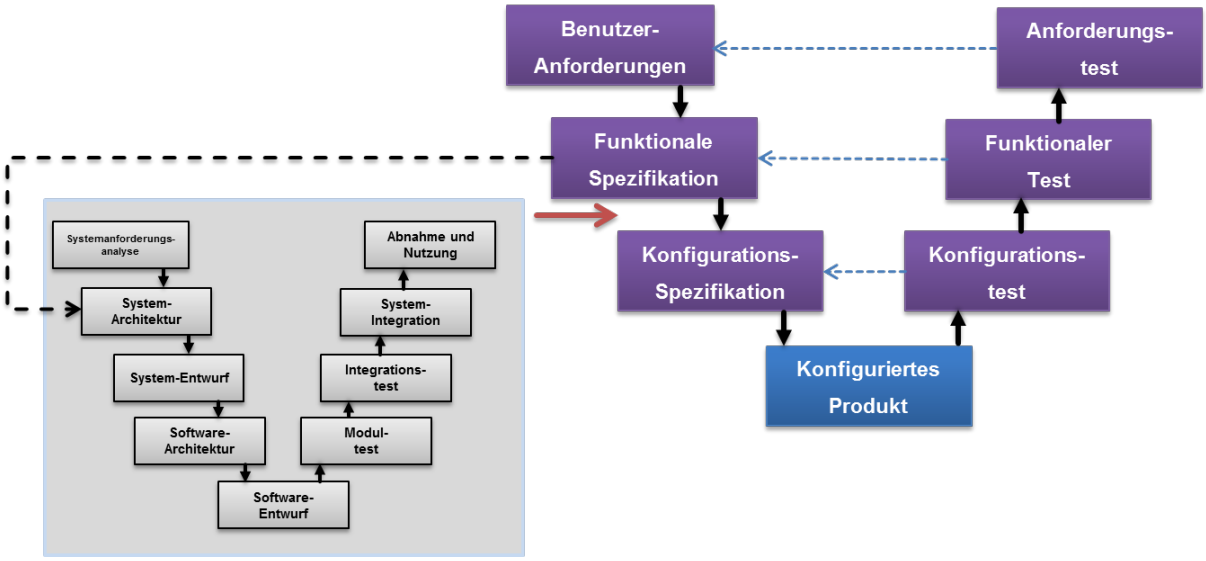


Abbildung 4.6: Das kaskadierte V-Modell

Parallel zur Entwicklung der virtuellen Komponenten sind nach diesem Vorbild, auch die Funktionsbausteine der Komponenten für die Steuerungssoftware zu entwickeln. In der folgenden Phase der Komponentenentwicklung sollen alle, für eine Komponente wichtigen Funktionalitäten, mit einer möglichst vollständigen Testabdeckung überprüft werden. An dieser Stelle werden alle Ein- und Ausgangssignale der Verhaltensbausteine der virtuellen

Komponente mit den Ein- und Ausgängen der entsprechenden Steuerungsbausteinen und der Kinematik eins zu eins gekoppelt. Dieser Integrationsschritt ist erneut zu testen. Der Vorteil dabei ist, dass in einer sehr frühen Entwicklungsphase alle internen Logik-, Diagnose- und Sicherheitsfunktionen in den virtuellen Baustein gegenüber der SPS bereits komplett durchgetestet werden, unabhängig von den Anwendungsszenarien, in denen sie später eingesetzt werden.

4.3.3.2 Das V-Modell für den Entwicklungsprozess der virtuellen Anlage

Das in Abschnitt 4.2.5 besprochene V-Modell nach GAMP 5 ist für die Entwicklung und Qualitätssicherung des realen automatisierten Systems gut geeignet und in den entsprechenden Branchen etabliert. Eine Analogie zu diesem V-Modell kann auch für virtuelle Anlagen angewendet werden. Für GAMP 5 wird davon ausgegangen, dass die Komponenten, die bei der Konfiguration der Anlage verwendet werden, qualitätsgesicherte reale Produkte sind. Dieser Ausgangspunkt gilt auch für die Entwicklung eines mechatronischen Anlagenmodells.

Entsprechend des V-Modells für konfigurierte Anlagen nach GAMP 5 müssen die notwendigen Komponenten schon vorhanden sein. Anhand der implementierten Virtualisierungs- und Steuerungsbausteine wird im nun folgenden Schritt der Integrationstest aller Komponenten zu einer Anlage durchgeführt. Besonders ist hier, dass nun entsprechend den Anforderungen an die Funktionalität der Anlage die Steuerungssoftware erstellt wird, in die die Funktionsbausteine der Komponenten eingegliedert werden. Die übrige Vorgehensweise ist entsprechend dem V-Modell nach GAMP 5.

4.4 Bestrebter Entwicklungsprozess

Basierend auf dem eingeführten kaskadierten V-Modell, wird ein neuer Entwicklungsprozess für die Modellierung und Validierung des mechatronischen Anlagenmodells für die virtuellen Inbetriebnahme vorgeschlagen.

4.4.1 Heutiger Geschäftsprozess für die Modellentwicklung

Heutzutage erfolgt der Entwicklungsprozess für die Simulationsmodellierung hauptsächlich folgende Schritte.

- Das erwünschte Systemverhalten wird durch einen OEM (Original Equipment Manufacturer) spezifiziert. Das Gesamtsystem wird in Teilsysteme (Komponenten) unterteilt.

- OEM entwickelt die Komponentenmodelle eigenständig. Alle gefertigten Komponentenmodelle werden in einer Projektbibliothek gespeichert. Manchmal wird dieser Schritt an einen Dienstleister beauftragt.
- Unterschiedliche Komponentenmodelle werden von OEM zu einem Anlagenmodell integriert.

Es ist zu sehen, dass der OEM allein für den kompletten Entwicklungsprozess der Simulationsmodellierung verantwortlich ist. Insbesondere ist die Erstellung der Komponentenmodelle keine triviale Aufgabe, wenn die vom OEM heutzutage alleine getragen wird. Da in der Regel nur die Hersteller der Komponenten das Wissen über die detaillierte interne Funktionsweise haben, ist es für andere schwierig bzw. unmöglich, das Verhalten von komplexeren Komponenten realitätsnah zu modellieren. Außerdem ist die Pflege der virtuellen Komponenten und Funktionsbausteine bei Weiterentwicklung der realen Komponenten für OEM sehr aufwendig.

Das OEM-Know-how über eine Komponente ist oft nur über die Dokumente wie Handbücher oder Datenblätter möglich, die mit der realen Komponente mitgeliefert werden und erklären nicht die Funktionalität dieser Komponente. Auf diese Weise ist das Wissen über die interne Funktionsweise einer Komponente nur sehr schwer oder gar unmöglich vollständig nachvollziehbar, was für den OEM zeit- und arbeitsaufwendig ist.

4.4.2 Notwendigkeit eines neuen Geschäftsprozesses

Es ist daher naheliegend, dass die Geräte- und Komponentenhersteller die benötigten Verhaltensmodelle für die virtuelle Inbetriebnahme erstellen und mit dem Produkt mitliefern. Dies ist z.B. schon heute bei den Geometriedaten in Form von CAD-Modellen oder den Kommunikationskonfigurationsdateien Stand der Technik. Die Bereitstellung der Verhaltensmodelle könnte eine ähnliche Entwicklung nehmen.

Dieser neue Prozess bedeutet erhebliche Vorteile sowohl für die OEMs als auch Komponentenhersteller.

Vorteile für Komponentenhersteller sind z.B.:

- höhere Qualität und Genauigkeit des Verhaltensmodells der Komponenten, da Komponentenhersteller, im Vergleich zu OEMs, mehr Know-how über ihre eigenen Produkte besitzen.
- ausführlichere Spezifikationen für die Testfallgenerierung wegen dem detaillierteren Know-how über die interne Struktur einer Komponente. Dies führt wiederum zur Erhöhung der Qualität und Robustheit des Komponentenmodells.

Mit dem neuen Prozess profitieren die OEMs durch:

- reduzierte Kosten durch ersparten Aufwand für die Einarbeitung in den fremden Technologien der Komponenten.
- wenigen Aufwand beim Integrationstest wegen der erhöhten Qualität der Komponentenmodelle.
- beschleunigter Entwicklungsprozess durch die neue Arbeitsverteilung zwischen OEMs und Komponentenhersteller, was auch eine beschleunigte „time to market“ bedeutet.

4.4.3 Neuer Geschäftsprozess nach dem kaskadierten V-Modell

Durch die getrennte Betrachtung für den Entwicklungsprozess auf Komponenten- und Anlagenebene eignet sich das in Abschnitt 4.3.3 eingeführte kaskadierte V-Modell besonders gut für die Beschreibung des neuen angestrebten Geschäftsprozesses.

Der Entwicklungsprozess der Komponentenentwicklung muss vor der Anlagenentwicklung erfolgen. Dies ist der Komponentenhersteller durchzuführen. In der Folge mehrerer Projekte wird eine Bibliothek virtueller Komponenten und Funktionsbausteine entstehen, die kontinuierlich gepflegt und nur bei Bedarf ergänzt werden muss.

Die Komponentenmodelle müssen so beschaffen sein, dass sie die Komponente nicht nur in einem speziellen Anwendungsfall abbilden, sondern alle späteren Nutzungsszenarien abdecken. Auf die Komponenten einer Anlage existieren in der Regel unterschiedliche Sichtweisen für die einzelnen Fachbereiche. Für jede dieser Sichtweisen gibt es eine Ausprägung der Komponente, welche das Modell abbilden muss. Das Komponentenmodell vereint alle diese Aspekte in sich, so dass je nach dem konkreten Anwendungsfall eine Ausprägung genutzt werden kann. Während zur Prüfung der mechanischen Funktionalität zum Beispiel Geometrie, Kinematik und auch Verhalten wichtig sind, benötigt man für die SPS-Programme die Aspekte Verhalten, Kommunikation sowie Visualisierung und Steuerungstechnik. Somit wird es möglich, ein zentrales Modell einer Komponente zu erstellen, das u.a. in Planung, Konstruktion und virtueller Inbetriebnahme genutzt werden kann. Diese Durchgängigkeit ermöglicht die Erstellung des Anlagenmodells für die virtuelle Inbetriebnahme parallel zur Konstruktion und Planung, da die notwendigen Informationen (Kinematik, Verhalten, etc.) bereits in diesen Schritten ins Modell eingefügt werden.

Komponentenentwicklung, Bibliothekspflege und Anlagenentwicklung werden mit den geschilderten Qualitätsmaßnahmen begleitet. Daraus entsteht ein durchgängig validierter Prozess. Dieser Entwicklungsprozess wird durch Abbildung 4.7 anschaulich dargestellt.

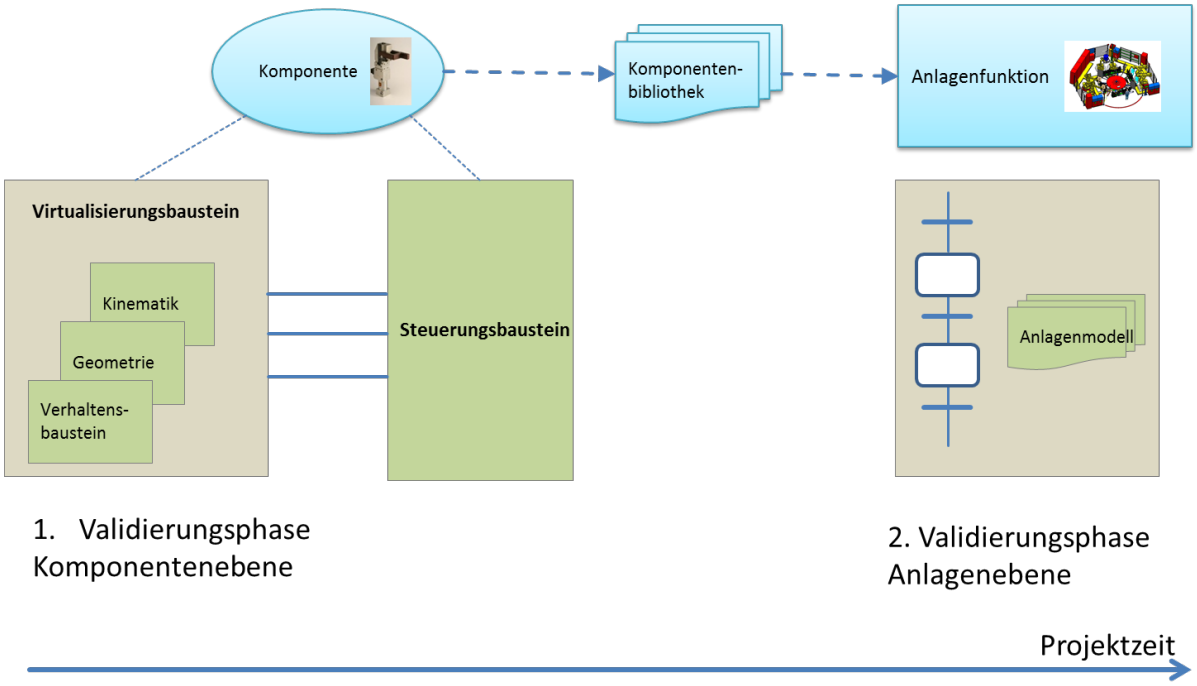


Abbildung 4.7: Das Validierungskonzept in zwei Schritte

Nach diesem Entwicklungsprozess werden die Verhaltensmodelle der einzelnen Komponenten für die Anwender der virtuellen Inbetriebnahme bereitgestellt, so dass nur noch die projektspezifischen Parameter zu den Bibliothekselementen angepasst werden müssen.

4.5 Zusammenfassung

In diesem Kapitel wird ein Vorgehensmodell für den Entwicklungsprozess virtueller Anlagen vorgeschlagen. Es wird zunächst ein Entstehungsprozess mit funktionaler Trennung nach anlagentechnischen und steuerungstechnischen Aspekten sowie einer strukturellen Trennung in Komponenten- und Anlagenebene vorgestellt. Das entsprechende Vorgehensmodell wird in Form eines kaskadierten V-Modells eingebettet, das auf den V-Modellen nach GAMP 5 und dem des Softwareengineerings basiert. Nach diesem Vorgehensmodell ist ein neuer Geschäftsprozess möglich, der auf der Komponentenentwicklung basiert. Im weiteren Teil dieser Arbeit wird auf die Modellierung der Komponente fokussiert.

5 Verhaltensmodellierung einer Komponente

Ein Verhaltensmodell simuliert das Anlagenverhalten gegenüber den Steuerungen durch die Abbildung des logischen und zeitlichen Verhaltens. Ein Simulationsmodell stellt ein Abbild der Wirklichkeit dar, bei dem nur die für virtuelle Inbetriebnahme wichtigen Eigenschaften in das Simulationsmodell übertragen werden.

Aus der Modellierungssicht lässt sich ein reales System aus unterschiedlichen Sichten darstellen. Da es niemals möglich ist, alle Aspekte eines realen Systems nachzubilden, wird es bei der Modellierung nur solche Eigenschaften berücksichtigt, die für die Anwendungsszenarien relevant sind. Z.B. für den Hallenplan sind die geometrischen Daten eines Förderbandes wichtig. Für die Berechnung der Taktzeit spielt jedoch die Transportgeschwindigkeit von demselben Förderband die wichtigste Rolle.

5.1 Verhaltensmodellierung für unterschiedliche Anwendungsszenarien

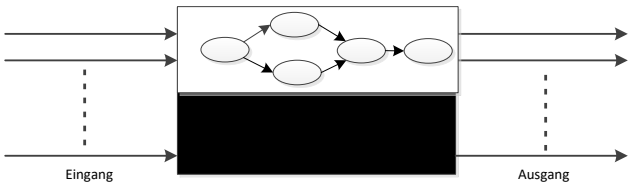
5.1.1 Modellierungsansätze

Wie in Abschnitt 2.1.2 bereits eingeführt, lässt sich die Modellierung eines Systems zwischen Black-Box-, Grey-Box- und White-Box-Ansatz unterscheiden.

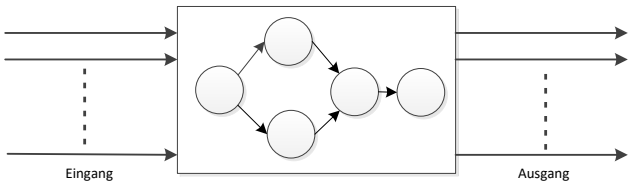
5.1.1.1 Black-Box-Modellierung

Ein zu untersuchendes System lässt sich als „Black-Box“ (vgl. Abbildung 5.1(a)) modellieren, wenn kein internes Wissen des Systems nach außen vorliegt. Ein Black-Box-Modell liefert den phänomenologischen Zusammenhang zwischen Ein- und Ausgang eines Systems ohne Kenntnis der zugrundeliegenden Gesetzmäßigkeiten. Ein solches Modell entsteht durch induktive Analyse und Beobachtung eines konkreten Systems ohne explizite Beschreibungen der wirkenden Prinzipien in dem Modell. Deswegen hat ein Black-Box-Modell

(a) Black-Box-Modellierung



(b) Grey-Box-Modellierung



(c) White-Box-Modellierung

Abbildung 5.1: Modellierungsansätze: Black-Box, Grey-Box und White-Box

normalerweise hohen empirischen Anteil, das ohne physikalische Einsicht deskriptiv ein System schildert. Aus diesem Grund ist es in der Regel schwierig, physikalische Bedeutung des Systems aus dem Modell selbsterklärend zu interpretieren.

Einfachste Beschreibungsmittel zur Darstellung eines Black-Box-Modells ist die Lookup-Tabelle (LuT), in der ein Satz numerischer Daten, die durch Beobachtung bzw. experimentelle Messung bestimmt und aufgenommen werden, um den funktionalen Zusammenhang des E/A-Verhaltens durch eine tabellarische Übersicht wiederzugeben. Für bestimmte Eingangsdatensätze aus der Lookup-Tabelle werden immer definiert Ausgangsdatensätze eindeutig zugeordnet. Durch solche wertediskrete Darstellung wird Black-Box-Modell durch indizierten Datenzugriff abgebildet.

Alternativ kann das Black-Box-Modell mit Hilfe der Methode Kurvenanpassung (oder Kurvenregression, *engl. curve fitting*) angenähert durch polynomiale Funktion beschrieben werden. Aufgezeichnete Messdaten als Grundlage der Modellbildung werden durch Polynom-Interpolation angepasst. Für eine polynomische Funktion in der Form

$$y = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n.$$

wird nach Regressionsparametern $(a_0, a_1, a_2..., a_n)$ gesucht, damit die Messdaten am besten angepasst werden. Dafür können unterschiedliche Verfahren eingesetzt werden, wie z.B. die Methode der kleinsten Quadrate, mit der die Quadratssumme der Differenz zwischen Messdaten und Funktionswerten minimal gehalten wird.

Die approximierte Funktion kann linear sein, in der nur a_0 und a_1 als Parameter ausgewertet werden. Es kann allerdings auch nichtlineare Funktion mit der Ordnung n bestimmt werden, indem $n + 1$ Anpassungskoeffizienten $a_0, a_1, a_2..., a_n$ zu abschätzen sind.

Größter Vorteil der Black-Box-Modellierung besteht in den geringen Bedarf an Rechenleistung wegen der einfachen Modellstruktur, dadurch lässt sich ein Black-Box-Modell schnell in Echtzeit simulieren. Der Aufwand der Modellerstellung mit den empirischen Ansätzen ist im Übrigen relativ gering, da ein internes Verständnis zu dem System nicht nötig ist.

Nachteilig ist die fehlende Flexibilität. Zum einen ist die begrenzte Möglichkeit für die Extrapolation, die prognostiziert, wie das Modell sich außerhalb des betrachteten Bereichs verhält. Da die Daten zur Ermittlung der approximierten Kurve nur in einen gewissen Bereich gültig sind. Zum anderen müssen die Daten erneut gemessen werden, wenn das System sich physikalisch ändert. Des Weiteren bietet dieser Ansatz wenige Detailtiefe, da hier nur das Ein- und Ausgangsverhalten abgebildet wird.

5.1.1.2 White-Box-Modellierung

Im Gegensatz zum Black-Box-Modell werden White-Box-Modelle (vgl. Abbildung 5.1(c)) deduktiv aus Analyse abgeleitet. Die Konstruktionen sowie funktionalen Zusammenhänge eines Systems werden ermittelt und anschließend durch naturwissenschaftlich-mathematische Gesetze bestimmt. Voraussetzung dafür ist, dass die innere Struktur und derer Wirkungszusammenhänge eines Systems bekannt sind. Detaillierte Innenstruktur sowie physikalischer Zusammenhang eines Systems werden durch White-Box-Modell abgebildet. In dieser Dissertation werden White-Box-Modell und physikbasiertes Modell als Gleiche betrachtet.

Es ist nicht nötig und fast unmöglich, alle Aspekte eines Systems zu analysieren und durch White-Box-Modell zu beschreiben. Die Detaillierung des Modells ist oftmals von den Anwendungsanforderungen bestimmt.

Bei der White-Box-Modellierung werden physikalische Gesetze durch Differentialgleichungen beschrieben [Güh10], die aus A-Priori-Informationen, d.h. theoretischen Kenntnissen zu dem System abgeleitet werden.

Zu den Vorteilen der White-Box-Modellierung gehören die hohe Detaillierungsstufe sowie Flexibilität. Da das Modell selbst auf der physikalischen Einsicht eines Systems basiert, ist das Modell für einen breiten Arbeitsbereich gültig. Auch bei geänderten Umgebungsbedingungen ist es nicht nötig, Messungen erneut durchzuführen. Bei bestimmten Anregungen auf das System zeigt das White-Box-Modell nicht nur die Reaktionen, sondern auch das innere Wirkprinzip, das aufweist, wie das von außen beobachteten Verhalten im Systeminneren erzeugt wird.

Hauptnachteil eines White-Box-Modell besteht in der hohen Modellkomplexität. Da diese Modellart laut Definition kaum Modellvereinfachungen zulässt, wäre großer Aufwand bei der Modellerstellung sowie große Rechenleistungsbedarf notwendig. Für virtuelle Inbetriebnahme kann die Simulation zudem in Echtzeit nicht garantiert werden.

5.1.1.3 Grey-Box-Modellierung

Bei Grey-Box-Modellierung handelt sich um eine Mischform von Black-Box- und White-Box-Modellierung, die die Vorteile der beiden letzten Ansätze kombiniert. Ein Grey-Box-Modell besteht zum Teil aus White-Box-Modell, bei dem alle bekannten Informationen zum System abgebildet werden. Die Restliche, wo Systemkenntnis fehlt oder unsicher ist, wird durch Black-Box-Modelle approximiert modelliert (vgl. Abbildung 5.1(b)).

Da es in den meisten industriellen Anwendungen eingeschränkt ist, vollständige Kenntnisse über System zu gewinnen, bietet Grey-Box-Modell große Flexibilität bei der

Modellierung. Als ein guter Kompromiss zwischen Black-Box- und White-Box-Modell, kann Grey-Box-Modell deutlich detailliertere Information als Black-Box-Modell liefern. Gleichzeitig bleibt die Modellkomplexität auf ein relativ überschaubares Niveau. Verhältnis des Anteils von Black-Box und White-Box ist abhängig davon, wieviel Systeminformation zur Verfügung gestellt wird. Selbst wenn die Struktur eines Systems teilweise bekannt ist, sind Systembeschreibungen durch genaue Funktionen zu den einzelnen Teilsystemen möglicherweise unumgänglich ist. In diesem Fall kann Grey-Box-Modell die verfügbaren Informationen ausnutzen, unbekannte Informationen lassen sich experimentell identifizieren. Somit ist Grey-Box-Modell in Abhängigkeit von den Anforderungen in unterschiedlichem Detaillierungsgrad skalierbar.

5.1.1.4 Bewertung

	Black-Box	**Grey-Box**	**White-Box**
Detaillierungs-stufe	niedrig	mittel	hoch
Modell-komplexität	niedrig	mittel	hoch
Rechenleistungs-bedarf	niedrig	niedrig/mittel	mittel/hoch
Modellierungs-aufwand	niedrig	niedrig/mittel	hoch
Flexibilität	niedrig	mittel/hoch	hoch

Tabelle 5.1: Bewertung der unterschiedlichen Modellierungsansätze

Die drei Modellierungsarten werden in Tabelle 5.1 zusammengefasst bewertet. Es ist zu sehen, dass Black-Box-, Grey-Box- und White-Box-Modell eine steigende Detaillierungsstufe sowie Modellkomplexität haben. Der gebrauchte Rechenleistungsbedarf sowie Modellierungsaufwand werden dementsprechend größer. Gleichzeitig bietet ein White-Box-Modell jedoch höhere Flexibilität gegenüber den beiden anderen Modellansätzen. Auswahl der richtigen Modellierungsart ist meistens ein Kompromiss zwischen Anwendungsbedarf und verfügbaren Ressourcen. Für die meisten Anwendungsfälle in der virtuellen Inbetriebnahme werden Black-Box- und Grey-Box-Modelle eingesetzt. White-Box-Modelle sind zwar sehr genau und realitätstreu, sind aber wegen dem hohen Aufwand in den Erstellungsprozess und hohem Ressourcenbedarf bei der Modellausführung auf einen Rechner nach dem heutigen Stand der Technologie nur flächendeckend in der Industrie einsetzbar.

5.1.2 Modellierung am Beispiel einer Zylinder-Ventil-Kombination für unterschiedliche Anwendungsszenarien

Im Folgenden werden die in letzten Abschnitt eingeführten Modellierungsansätze mittels einer hydraulischen Zylinder-Ventil-Kombination beispielhaft erklärt, die im Wesentlichen aus einem 5/2-Wege-Impulsventil sowie einen doppeltwirkenden Arbeitszylinder besteht [Bau05]. Ein 5/2-Wegeventil (Abbildung 5.2) hat 5 gesteuerte Anschlüsse sowie 2 Schaltstellungen, welches um Freigabe oder Sperrung des Durchflusses sowie Änderung der Durchstromrichtung sorgt. Als Stellglied steuert das Wegeventil den Ölstrom in der gewünschten Richtung an. Das Öl übt Druck auf den Kolben aus, der zu dem angeschlossenen Arbeitszylinder gehört. Durch den Arbeitszylinder wird hydraulische Energie in eine translatorische, mechanische Kraft und Bewegung umgewandelt. Ausfahrtrichtung des Zylinders wird von der Stellung des Wegeventils bestimmt.

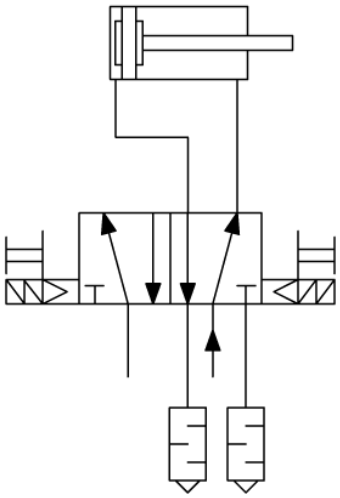


Abbildung 5.2: Vereinfachte Darstellung einer Zylinder-Ventil-Kombination

5.1.2.1 Simulationsmodell für Austaktung

In der Fertigungsindustrie ist es sehr üblich mit einer Fertigungslinie unterschiedliche Produktvarianten zu produzieren. Aus diesem Grund ist es wichtig, für jede Variante einen optimierten Fertigungsprozess zu definieren. Austaktung beschreibt den Planungsprozess zur Auslegung getakteter Fertigungslinien. Dies erfolgt durch die Beschreibung der Prozessschritte mit ihren Tätigkeiten und der dazugehörigen Arbeitszeit. Dadurch wird die Arbeitsfolge unter Berücksichtigung der Taktzeit bestimmt. Taktzeit beschreibt die Zeit, die ein zu produzierendes Werkstück für einen Prozessschritt verbrauchen soll.

Austaktung findet normalerweise in einer frühen Phase der Anlagenplanung statt, bei der noch keine realen Anlagen vorhanden sind. Da zu dieser Zeit nur wenige technischen Details

zu den einzelnen einzusetzenden mechatronischen Komponenten in einer Fertigungsanlage bekannt sind, steht meistens nur ein Taktzeitdiagramm zur Verfügung (siehe Abbildung 5.3), in dem die einzelnen Arbeitsabläufe sowie deren entsprechend vorgesehenen Zeit in einem Gantt-Diagramm dargestellt wird. Es ist zu sehen, dass in einem Taktdiagramm die Arbeitsvorgänge auf unterschiedliche Arbeitsstationen verteilt werden, die zeitlich parallel operieren können. Zu jeder Arbeitsstation wird Taktzeit zugeordnet, die die maximale Bearbeitungszeit eines Werkstücks in einer Station festlegt. Z.B. für eine Förderstrecke in der Logikplanung, die aus mehreren Stationen besteht, lässt sich die Zeitdauer für den gesamten Verfahrweg aus den Zeitdauern für die einzelne Station ermitteln, die parallel oder in Reihe miteinander verbunden werden. Durch eine optimierte Gestaltung der Austaktung lässt sich somit die wertschöpfenden Anteile im Fertigungsprozess steigern und eine hohe Auslastung der Anlage ermöglichen.

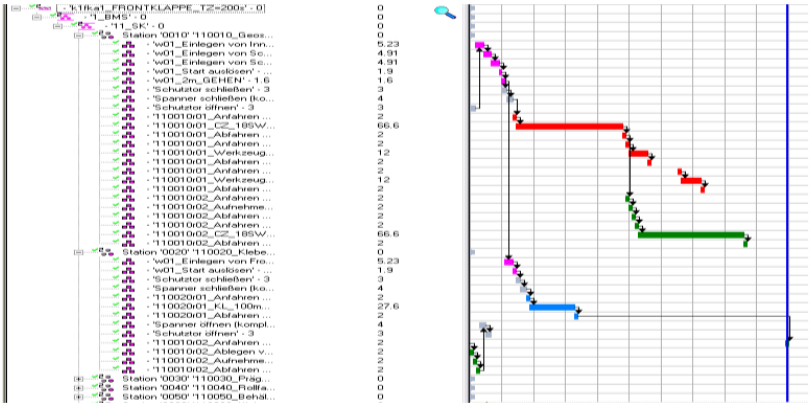


Abbildung 5.3: Taktzeitdiagramm als Ausgangspunkt des Prozessplans

Es ist sinnvoll, Simulationsmodell zu erstellen, um den Prozessplan der Austaktung zu überprüfen und optimieren. Da Taktzeit hier die wichtigsten Größen in Bezug auf der Austaktung ist, lässt sich ein solches Simulationsmodell relativ trivial erstellen. Simulationstechnisch kann die Zeitdauer eines bestimmten Vorgangs mittels eines Totzeitgliedes dargestellt werden. Hierbei ist die interne Struktur irrelevant, da hier Zeit der einzige Parameter ist. Abbildung 5.4 zeigt beispielhaft Modellierung der Zylinder-Ventil-Kombination mit Totzeitmodell als Blackbox-Ansatz.

Es ist zu sehen, dass die Bewegungen der Vor- und Rückfahrt jeweils durch ein Totzeitglied modelliert werden. Da hier keine interne Struktur in dem Simulationsmodell behandelt wird, gehört solches Totzeitmodell für Austaktung offensichtlich zur Kategorie des Black-Box-Modells. Durch logische Verbindungen unterschiedlicher Totzeitglieder kann die gesamte Zeitdauer zwischen Quellen und Senken einer Strecke simulativ bestimmt werden.

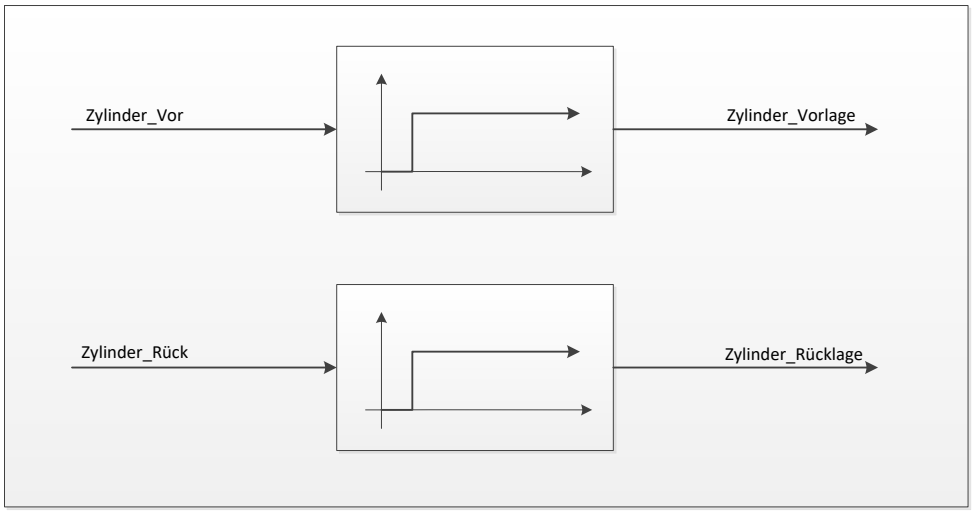


Abbildung 5.4: Modellierung nach Totzeit

5.1.2.2 Simulationsmodell für Steuerungstest

Wie in Abschnitt 2.3 eingeführt, besteht ein wichtiges Anwendungsgebiet der virtuellen Inbetriebnahme in frühzeitigen Steuerungstest anhand Simulationsmodelle. Durch Kopplung des Simulationsmodells mit einer SPS lässt sich SPS-Programm testen, bevor eine reale Komponente bzw. Anlage bestellt wird. Es stellt Anforderung an das Simulationsmodell, dass das Modell auf die Befehle (Steuerungsausgangssignale) der Steuerung reagieren und gleichzeitig entsprechende logischen und zeitlichen Rückmeldungen (Steuerungseingangssignale) zurückgeben kann. Diese Anforderung kann ein reines Totzeitmodell aus dem letzten Abschnitt in den meisten Fällen nicht bewerkstelligen. Jedoch lässt sich die Modellkomplexität nicht beliebig erhöhen wegen der Rechenkapazität der gesamten Simulationsumgebung. Insbesondere für große Anlagen, die aus mehreren hundert Komponenten bestehen, ist die verfügbare Rechenleistung für jede Komponente nochmals stark begrenzt.

Eine typische Steuerungsaufgabe ist die Positionierungsregelung. Die Bewegung eines motorgetriebenen Objekts wird von der Steuerung geregelt anhand der aktuellen Position. Einreichung einer bestimmten Position bedeutet z.B. Zustandswechsel in der Steuerung. Dafür ist die Positionsbildung in Abhängigkeit von Zeit für die Modellierung relevant. Wenn neben Positionswerten keine weiteren Zustandsgrößen erforderlich sind, ist Black-Box-Modellierung hier einsetzbar, indem die Positionen im Zeitverlauf direkt abgebildet werden. Solche Verhalten lassen sich meistens durch einfache Signalübertragungsglieder wie z.B. Integratoren, PT1- und PT2-Glieder beschreiben.

Abbildung 5.5 zeigt ein Modell der genannten Zylinder-Ventil-Kombination mit dem Blackbox-Ansatz. Zusätzlich zu dem Totzeitmodell aus Abschnitt 5.1.2.1 erhält neben Zeit noch Information über die aktuelle Position, die für viele Anwendungsszenarien notwendig ist. Hier wird vereinfacht angenommen, dass der Zylinderkolben zwischen den

beiden Grenzpositionen stets mit konstanter Geschwindigkeit ein- und ausfährt. Diese Annahme ist zwar sehr grob, ist für meiste Anwendungen in der VIBN, z.B. Darstellung der Positionsänderung in der Visualisierung, detailliert genug.

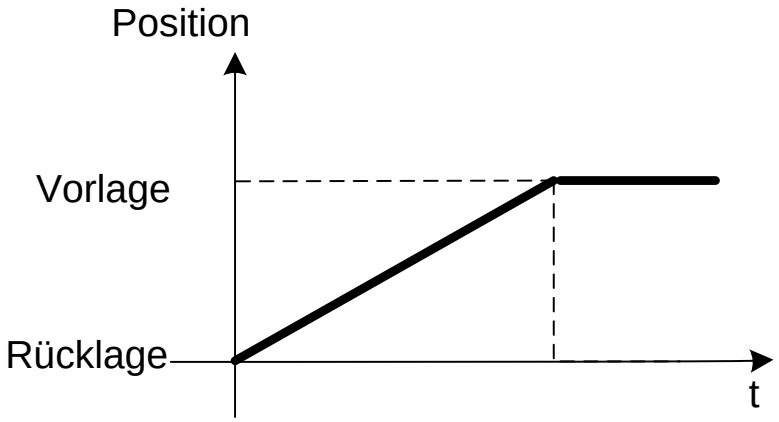


Abbildung 5.5: Blackbox-Modellierung Ebene 2

Für manche Anwendungen sind neben Position und Zeit andere Zustandsgröße für bestimmte Steuerungsaufgaben von Bedeutung, z.B. Durchfluss aus Regelventil oder Kolbengeschwindigkeit des Zylinders. Ermittlung solcher Größe setzt bestimmte Kenntnisse über Systemstruktur voraus. Hierfür bietet sich Grey-Box-Modell gut an. Im Gegensatz zum Black-Box-Ansatz, in dem nur ein allgemeines Position-Zeit-Verhältnis dargestellt wird, besteht Grey-Box-Modell aus einen Wirkungsplan, in dem jede unterliegende Komponente aus einem oder mehreren Funktionsblöcken dargestellt wird. Jede Komponente lässt sich wiederum als Black-Box-Modell betrachtet werden, das das jeweilige Eingangs-/Ausgangsverhalten abbildet.

Für das Beispiel Zylinder-Ventil-Kombination, die aus Regelventil und Hydraulikzylinder besteht, ist die Wirkungsstruktur in Abbildung 5.6 zu ersehen. Für das Regelventil wird angenommen, dass es einen proportionalen Zusammenhang zwischen Eingangsspannung u und Durchfluss Q mit dem Faktor K_y besteht. Vereinfacht gilt es auch für den Hydraulikzylinder, dass die Fahrgeschwindigkeit mit dem Durchstrom aus Regelventil linear zunimmt. Die gesuchte Position ergibt sich aus der Integration der Geschwindigkeit über die Zeit.

Die beiden Modelle nach Abbildung 5.5 und Abbildung 5.6 können Informationen über das Position-Zeit-Verhältnis liefern. Zusätzlich im Grey-Box-Modell (Abbildung 5.6) sind Informationen wie Durchfluss bzw. Kolbengeschwindigkeit erhältlich, die Anwendern weitere Möglichkeiten bieten. Die beiden Modellierungsansätze liefern zwar keine Information über das physikalische Verhalten der Komponente, sind jedoch in meisten Fällen ausreichend detailliert gegenüber dem Steuerungstest.

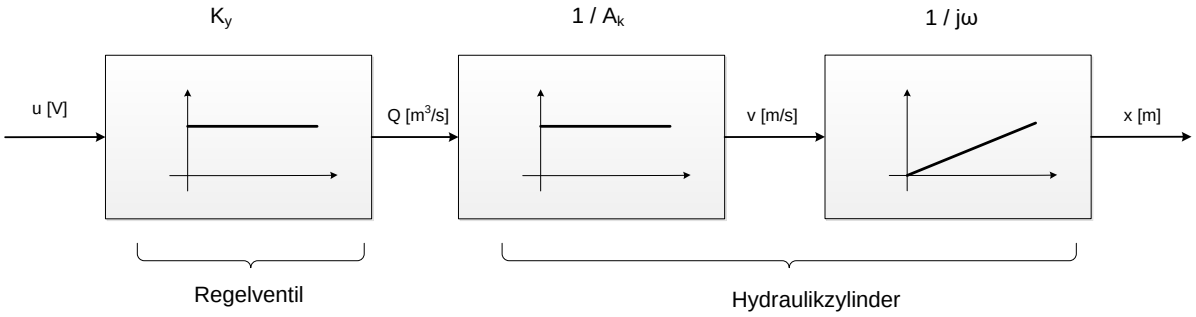


Abbildung 5.6: Whitebox-Modellierung Ebene 2

5.1.2.3 Simulationsmodell für detaillierte Analyse

Ein Whitebox-Modell hat die höchste Modellgenauigkeit. Bei dieser Detaillierungsstufe nähert sich das Systemabbild an das reale System an. Bei der Modellierung werden physikalische Effekte und dynamische Eigenschaften der Systeme berücksichtigt. Dies ist essenziell für viele systemnahe Analyse wie z.B. Einstellung der Regelparameter, bzw. Analyse der Kollisionseffekte.

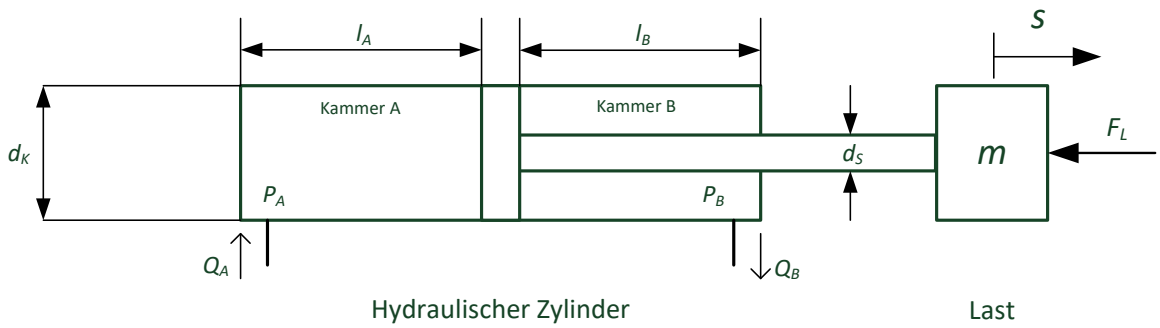


Abbildung 5.7: Schematische Darstellung eines hydraulischen Zylinders mit Last

Bei der Modellierung der Zylinder-Ventil-Kombination müssen die physikalischen Effekte wie hydraulische Kraft, dynamische Kraft berücksichtigt werden.

Die Bewegungsdifferentialgleichung des Kolbens lässt sich, vereinfacht halber unter Vernachlässigung der Reibungskraft, wie folgendes beschreiben:

$$m \cdot \ddot{S} = P_A \cdot A_A - P_B \cdot A_B - F_L = P_A \cdot \pi \cdot \left(\frac{d_k}{2}\right)^2 - P_B \cdot \pi \cdot \left[\left(\frac{d_K}{2}\right)^2 - \left(\frac{d_S}{2}\right)^2\right] - F_L \quad (5.1)$$

wobei

S : Zylinderkolbenposition
P_A : Druck in der Kammer A
P_B : Druck in der Kammer B
l_A : Länge der Kammer A
l_B : Länge der Kammer B
A_A : Kolbenfläche der Seite von Kammer A
A_B : Kolbenfläche der Seite von Kammer B
d_K : Kolbendurchmesser
d_S : Kolbenstangendurchmesser
Q_A : Volumenstromzufuhr
Q_B : Volumenstromabfuhr
F_L : äußere Belastung des Kolbens.

Der Zylinderraum besteht aus zwei Kammern, die durch einen Kolben getrennt werden. Die physikalischen Größen der beiden Kammern, wie z.B. Volumen und Druck sind abhängig von der Kolbenposition. Mittels einer richtigen Regelung der Zu- und Abfuhr des Volumenstroms lässt sich der Kolben in beiden Richtungen bewegen.

Da die Kolbenstange hier als Starrkörper betrachtet wird, gilt es hier:

$$\dot{l_A} = \dot{l_B} = \dot{S} \tag{5.2}$$

In der Hydraulik gibt Volumenstrom an, wie viel Volumen sich in einer gewisser Zeit durch einen Querschnitt bewegt. Somit lassen sich Volumenstrom Q_A und Q_B berechnen als:

$$\begin{aligned} Q_A &= \dot{V_A} = \dot{l_A} \cdot A_A = \dot{S} \cdot \pi \cdot \left(\frac{d_k}{2}\right)^2 \\ Q_B &= \dot{V_B} = \dot{l_B} \cdot A_B = \dot{S} \cdot \pi \cdot \left[\left(\frac{d_K}{2}\right)^2 - \left(\frac{d_S}{2}\right)^2\right] \end{aligned} \tag{5.3}$$

wobei hier bezeichnen V_A und V_B das jeweilige Volumen in Kammer A und Kammer B.

Durch Kombination der Formeln 5.1 und 5.3 kann z.B. ein Blockschaltbild des Zylinders entworfen werden, das die Abhängigkeit der verschiedenen physikalischen Größe

simulativ darstellen kann. Da die genaue Modellierung und Simulation der Zylinder-Ventil-Kombination nicht Gegenstand dieser Dissertation ist, sei bezüglich genaueren Details auf [Bes05] [WSG11] verwiesen.

5.1.2.4 Bewertung

Es ist zu entscheiden, welcher Modellierungsansatz die beabsichtigte Anforderung für das Testen von Steuerungsprogrammen erfüllt. Die Steuerprogramme sind so konzipiert, dass sie Abnormalitäten erkennen und angemessen reagieren, um ein sicheres Verhalten aufrechtzuerhalten. Eine Diagnose der Fehlerursachen ist in der Regel nicht Bestandteil des Programms, sondern Aufgabe zusätzlicher Serviceverfahren. Daher muss der Detaillierungsgrad der Modellierung so genau sein, so dass das Steuerprogramm richtig auf abnormales Verhalten reagieren kann. Für die virtuelle Inbetriebnahme passt der Grey-Box-Modellierungsansatz am besten, da er die erforderlichen Details für die Fehleridentifikation im Steuerungsprogramm modellieren kann. Tabelle 5.2 fasst die Auswahlkriterien verschiedener Modellierungsansätze nach den obigen Überlegungen zusammen.

	Black-Box	**Grey-Box**	**White-Box**
innere Struktur	unbekannt	teilweise unbekannt	bekannt
Ausgangssignal	durch Messung und Beobachtung	teils durch Messung/ Beobachtung, teils durch Annahme/ Annäherung	durch Annäherung/ mathematische Berechnung
innere körperliche Beziehung	unbekannt	teils angenähert, teils angenommen	bekannt und mathematische Berechnung

Tabelle 5.2: Auswahlkriterien verschiedener Modellierungsansätze

5.2 Modellierung des fehlerfreien Verhaltens

5.2.1 Modellierung der normalen Komponente

Eine Möglichkeit, die Anlage zu virtualisieren, stellt das mechatronische Anlagenmodell (vgl. Abbildung 5.8) dar, das grundsätzlich aus drei Bestandteilen besteht [KOB09]:

Verhaltensmodell ist das Herzstück des mechatronischen Anlagenmodells, welches das Verhaltensmodell, das Eingangs- und Ausgangsinterface und die interne Verhaltenslogik

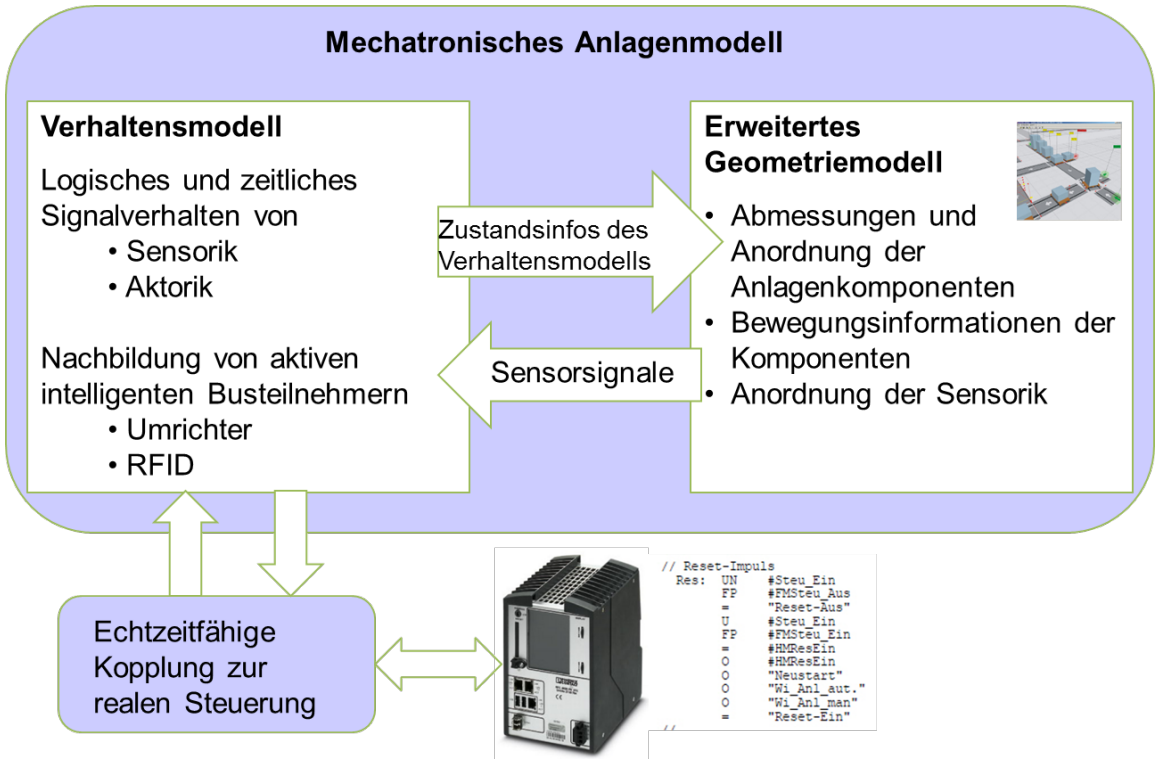


Abbildung 5.8: Aufbau und Bestandteile eines VIBN-Arbeitsplatzes (in Anlehnung an [KOB09])

widerspiegelt. Ein Verhaltensmodell ist üblicherweise aus verschiedenen Bausteinen aufgebaut, die das Verhalten einzelner Komponenten repräsentieren. Das E/A-Interface ermöglicht es, dass die virtuellen Komponenten signaltechnisch von einer SPS angesteuert werden können, während durch die Verhaltenssimulation das logische und zeitliche Verhalten des einfachen Betriebsmittels gegenüber der angeschlossenen Steuerung beschrieben wird. Für Betriebsmittel mit komplexeren technischen Funktionen wie Umrichter oder Identsystem z.B. RFID-Model, müssen auch die intelligenten Steuerungsfunktionen bei der Modellierung des Komponentenmodells berücksichtigt werden.

Erweitertes Geometriemodell sorgt für die Visualisierung des Anlagenmodells. Dabei werden die CAD-Daten aus der Konstruktion um kinematische Informationen erweitert. Auf der einen Seite sorgt das erweiterte Geometriemodell für die Visualisierung der Bewegungsabläufe des Anlagenmodells, auf der anderen Seite wird die Rückmeldung an das Verhaltensmodell durch die integrierte Sensorik z.B. Lichtschranke oder Endlagenschalter erzeugt. Schnittstellen müssen vom erweiterten 3D-Geometriemodells zum Verhaltensmodell bereitgestellt werden. Eine direkte Schnittstelle mit den Steuerungsprogrammen ist optional, da die Kommunikation zu diesen Programmen auch über das Verhaltensmodell erfolgen kann.

Bus-Emulation ermöglicht die Kommunikation der Steuerung mit der realen Anlage. Dabei wird das Kommunikationsverhalten aller Busteilnehmer emuliert. Zur virtuellen Inbetriebnahme kann dazu sowohl der reale Bus (z. B. Profibus oder Profinet) als auch andere Schnittstellen, wie beispielsweise OPC oder API, verwendet werden. Der Vorteil der Verwendung des realen Bussystems ist, dass die Steuerungsprogramme samt den Projekteinstellungen unverändert bleiben können und sich für den Softwaretester kein Unterschied zu einer realen IBN darstellt. Somit ermöglicht die VIBN ein frühzeitiges Testen der Steuerungsprogramme, bevor die reale Anlage verfügbar ist.

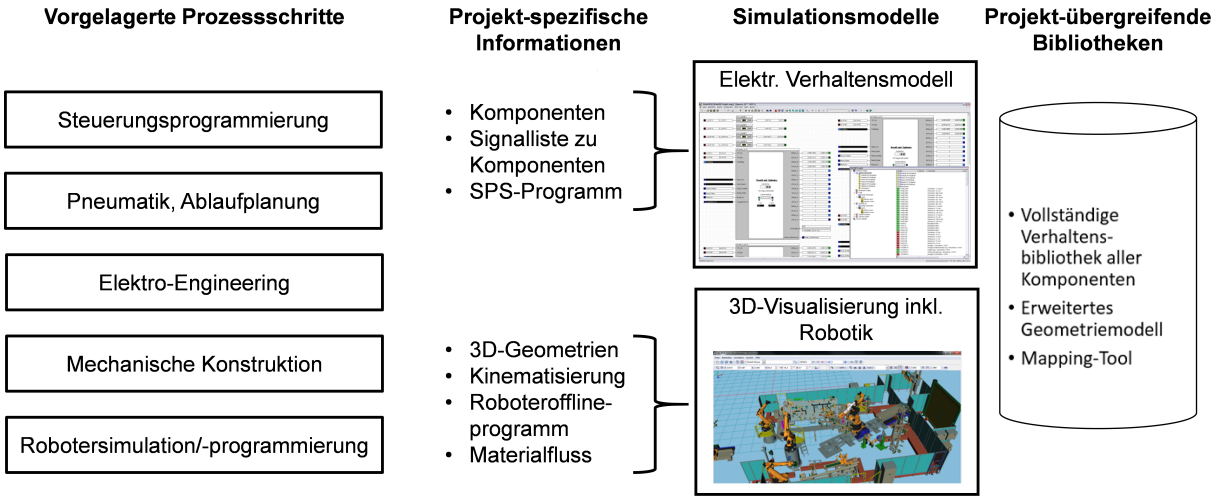


Abbildung 5.9: Entwicklungsprozess des Simulationsmodells

Da das Verhalten einer gesamten Anlage sehr komplex ist, ist es sinnvoll, das gesamte mechatronische Anlagenmodell aus einzelnen Komponenten hierarchisch aufzubauen. Dazu müssen alle Komponenten, die benötigt werden, in einer projektneutralen Komponenten-Modellbibliothek zur Verfügung stehen. Dieser Ansatz ermöglicht die Wiederverwendbarkeit der Bibliothekselemente sowohl projektintern als auch projektübergreifend.

Damit ein Komponentenmodell sich wie die reale Komponente bei der Simulation verhält, werden dafür zahlreiche Informationen, vor allem Planungsergebnisse aus vorgelagerten Planungsprozessschritten gebraucht. Diese Informationen sind vielfältig, voneinander abhängig und umfassen viele Engineeringdisziplinen (Mechanische Konstruktion, Elektro-Engineering, Pneumatik sowie Steuerungs- und Roboterprogrammierung).

Zusätzlich werden zur Erstellung des anlagenbezogenen Gesamtmodells projektspezifische Informationen gebraucht. Für das Anlagenverhaltensmodell werden Komponenten- und Signalliste gebraucht, die angeben, welche SPS-Signale mit den jeweiligen Komponenten zu verbinden sind.

Für das erweiterte Geometriemodell dienen hauptsächlich die kinematisierten CAD-Daten als Basis. Für eine Anlage mit logistischen Teilen sind Informationen aus Materialfluss

essentiell. Falls die Anlage Roboter enthalten, sind die Roboterofflineprogramme (OLP) aus der Robotersimulation notwendig.

Es hat sich herausgestellt, dass besonders die Ergebniszusammenführung der verschiedenen Engineeringdisziplinen aufwandsintensiv ist (so genanntes Mapping). Somit ist ein Mapping-Tool an dieser Stelle ein wichtiges Hilfsmittel für die Zusammenstellung aller Modelle.

5.2.2 Modellierung der intelligenten Komponente

Hintergrund ist die hierarchische Struktur in den automatisierungstechnischen Lösungen (Abbildung 5.10). Die Basisautomatisierungsfunktionen steuern z.B. einzelne Aktoren oder auch komplexere Messumformer an. Sie gewährleisten zusätzlich zu der operativen Steuerung weiterhin auch die Funktionsansteuerung mit Behandlung von abnormalen Verhalten, z.B. Abschaltung oder Verriegelung, sowie die Vorverarbeitung von Status- und Diagnosedaten. In der nächsten Hierarchieebene - in der Leitebene - befinden sich die ablauforientierten oder koordinierten Funktionen aufbauend auf den Basisfunktionen. Oberhalb der Leitebene, in der MES/ERP-Ebene befinden sich die Funktionen, die nicht mit dem direkten Prozess zu tun haben, wie z.B. Produktionsplanung, -analyse und -optimierung.

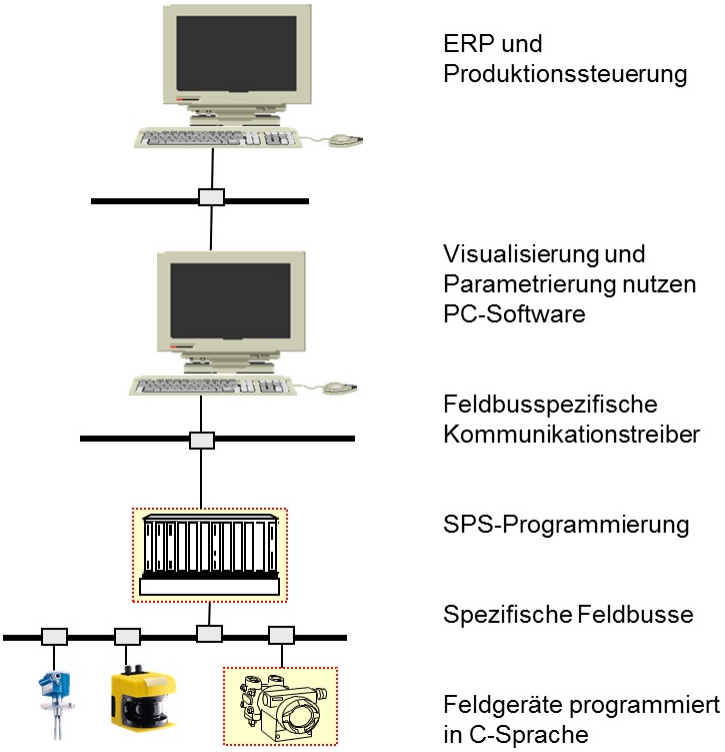


Abbildung 5.10: Allgemeine Hierarchie eines Automatisierungssystems

Hierbei ist die Tendenz für viele Komponenten wie Frequenzumrichter bzw. Identifikationsgeräte feststellbar, Basisautomatisierungsfunktionen weiter in die Feldkomponente

zu verlagern. Diese Dezentralisierung bietet folgende Vorteile im Vergleich zu einer traditionellen Lösung:

- Es wird ein innerer Regelkreis abgebildet, der möglichst prozessnah arbeiten und schnell reagieren kann.
- Hoher Modularisierungsgrad verbessert die Wiederverwendungsfähigkeit.
- Die aufwendige Verkabelung lässt sich durch einfache Busleitung ersetzen. Zusätzlich wird das Risiko von Verdrahtungsfehlern sowie verbundener Aufwand für die Fehlersuche minimiert.
- Die übergeordnete SPS wird von zahlreichen Steuerungsaufgaben in Echtzeit entlastet, da viele prozessnahen Funktionen in den Feldkomponenten ausgeführt werden und nur in der Steuerung über Funktionsbausteine (Proxies) auszulösen bzw. zu kontrollieren sind.

Solche Komponenten werden in dieser Arbeit als intelligente Komponenten benannt, die von „traditionellen“ Komponenten sich dadurch, dass technologische Funktionalitäten durch die aus SPS verlagerte integrierte Steuerungsfunktion unterscheiden.

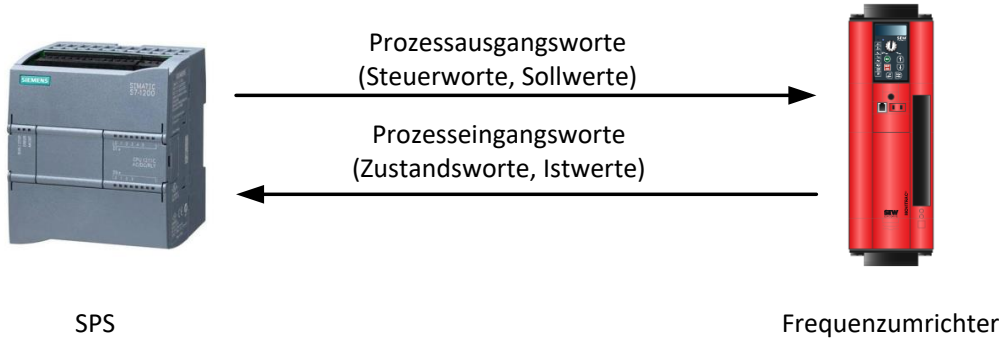


Abbildung 5.11: Datenaustausch zwischen SPS und Frequenzumrichter

Abbildung 5.11 stellt die Kommunikation zwischen einer SPS und einem Frequenzumrichter dar, die auf PROFIBUS-DP oder andere Feldbuskommunikationsstandards beruht. Die Befehle von der übergeordneten SPS für den Antrieb, wie z.B. Freigabe-Signal, Betriebsarten, Sollwerte usw., werden in Form von Prozessausgangsworten von SPS auf Frequenzumrichter übermittelt. Die Istwerte sowie andere Statusmeldungen und Fehlerdiagnosesignale werden als Zustandsworte von Frequenzumrichter an die SPS geleitet. Die eigentliche Regelung, von einfacher Drehzahlregelung bis hin zur komplexen synchronisierten Mehrachsenpositionierung findet antriebsnah intern im Frequenzumrichter statt.

Im Wesentlichen besteht eine intelligente Komponente aus folgenden Teilen:

Steuerungsschnittstelle

Die Kommunikation zwischen einer intelligenten Komponente und der Steuerung erfolgt über die Prozessdatenworte. Sie werden aus der Sicht von der SPS als Prozessausgangsworte (PAW) und Prozesseingangsworte (PEW) benannt, welche für die intelligente Komponente jeweils genau umgekehrt Eingang und Ausgang darstellen.

Damit PAW für die weitere Verarbeitung interpretierbar sind, müssen sie zunächst in einzelnen Bits aufgelöst werden. Je nach Signaltyp lassen sich ein Bit oder mehrere Bits auf neue Signale abbilden. Das Gleiche gilt für PEW, die fassen alle Daten wieder zusammen, die zurück an SPS zu übertragen sind.

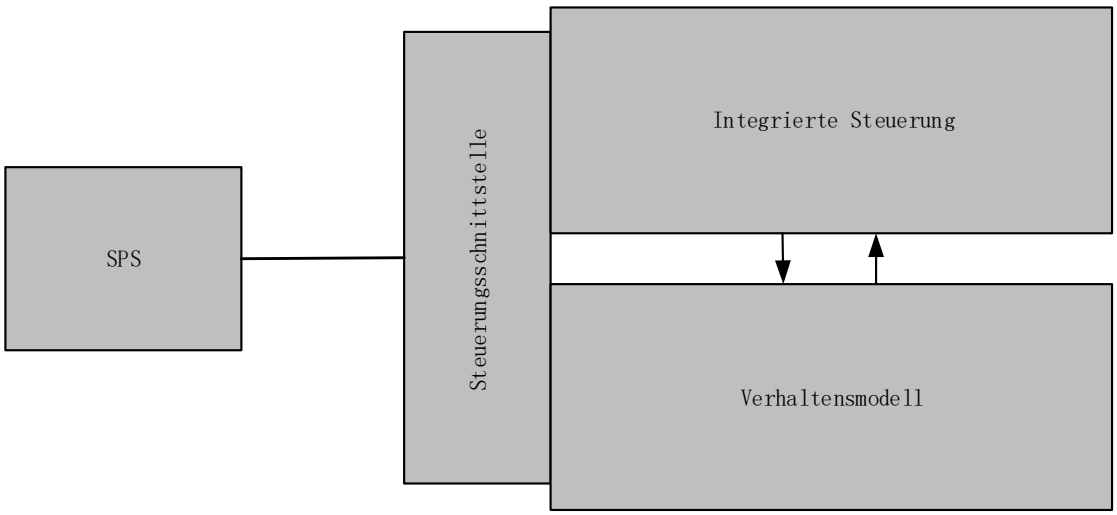


Abbildung 5.12: Strukturelle Darstellung einer intelligenten Komponente

Integrierte Steuerung

Im Vergleich zu den intelligenten Komponenten sind herkömmliche Komponente nicht mit technologischen Funktionen ausgerüstet und „nur“ parametrierbar und konfigurierbar. Der wesentliche Unterschied besteht in das integrierte Steuerungsteil.

- Abbildung der Betriebsarten: Viele intelligenten Komponenten können in verschiedene Betriebsarten arbeiten wie Automatik-, Tippbetrieb oder andere Betriebsmodi je nach Anwendungsfällen. Auswahl der Betriebsarten erfolgt meistens durch bestimmte Bits in Steuerworte, die unterschiedliche Betriebsmodi codiert repräsentieren.
- Abbildung der Ablaufsteuerung: Jede Betriebsart bzw. Anwendung hat einen Ablauf, der aus mehreren aufeinanderfolgenden Schritten besteht und schrittweise ausgeführt werden soll. Der Übergang von einem Schritt auf den folgenden erfolgt entsprechend

der Übergangsbedingung. Ablaufsteuerung dient als Koordinator zwischen der übergeordnet SPS und der zugeordnet gesteuerten Komponente. Neben der Echtzeitsteuerung werden auch alle prozessrelevanten Daten kontinuierlich überwacht und analysiert. Störungen werden durch Überwachung der Lastsituation erkannt und durch Einführung anwendungsspezifischer Maßnahmen behandelt. In vielen Fällen lässt sich der Normalbetrieb autark durch die intelligente Komponente herstellen. Die Implementierung der Ablaufsteuerung entspricht dem Sequential Function Chart (SFC) nach IEC 61131-3 und wird in dieser Arbeit nicht weiter behandelt.

Verhaltensmodellierung

Das Verhalten des Modells unterliegt der integrierten Steuerung. Das Prinzip der Modellierung ist vergleichbar mit der Modellierung einer normalen Komponente (siehe Abschnitt 5.2.1). Der Detaillierungsgrad des Modells wird durch den Umfang und der Struktur der integrierten Steuerung bestimmt.

5.3 Modellierung des Störverhaltens

Eine grundsätzliche Anforderung an industrielle Anlagen ist, dass diese Anlage nicht nur auf normale sondern auch auf gestörte Situation sicher und fehlerfrei reagiert. D.h. bei Störung oder fehlerhafter Bedienung soll diese Anlage in der Lage sein, die Anlage durch die Steuerung in einen gefahrlosen Zustand zu bringen. Beispielsweise bei einem verklemmten Motor soll der Schutzschalter getätigt werden, um einen Motorschaden zu vermeiden.

Dasselbe Prinzip gilt auch für die Simulation einer solchen Anlage. Es ist nicht ausreichend, bei der Modellierung nur das fehlerfreie Verhalten, das die intendierte Funktion gegenüber der Spezifikation beschreibt, abzubilden, was in Abschnitt 5.2 bereits behandelt wird. Gleich wichtig ist das Störverhalten zu berücksichtigen, das simuliert, wie ein System sich in einen fehlerbehafteten Zustand verhält.

5.3.1 Störverhalten zum Test der Fehlermeldung

Virtuelle Inbetriebnahme wird momentan vorrangig dazu genutzt, um die Logik des Steuerungsprogramms der SPS zu testen. Man überprüft die Logik des Steuerungsprogramms dahingehend, ob die geforderten Funktionen der Anlage bzw. deren Komponenten wie gewünscht ausgeführt werden [LSC12]. Während der Überprüfung geht man davon aus, dass das Simulationsmodell der Anlage fehlerfrei ist. Wenn das Simulationsmodell

der Anlage die gewünschte Funktionsweise ausführt, kann man schlussfolgern, dass das Steuerungsprogramm richtig arbeitet.

Neben der Behandlung von fehlerfreien Verhalten sind im Steuerungsprogramm auch Bestandteile vorhanden, welche ausgehend von den unerwünschten Fehlerfällen der Anlage bzw. der Anlagenkomponenten Fehlermeldungen ans Leitsystem generieren und potentielle Maßnahmen als Reaktion an die Anlagenkomponenten auf Fehlerfällen einleiten. Solche Fehlermeldungen werden zur Visualisierung weitergeleitet, die dann den Anlagenbediener durch textuelle bzw. grafische Darstellung über die Fehlerursache informiert. Sind die Fehler nicht sicherheitsrelevant, wird eine Warnung generiert. Wenn es um sicherheitskritische Fehler geht, wird eine Störung gemeldet. Erst wenn die Fehlerquelle beseitigt und die Störung quittiert ist, dürfen die weiteren Prozesse der Anlage durchgeführt werden.

Heutzutage können die meisten Simulationsmodelle nur das fehlerfreie Verhalten der Anlage bzw. Komponente darstellen. Das Testen des fehlerorientierten Verhaltens ist für einen Tester schwierig, da er normalerweise keine internen Kenntnisse von dem Simulationsmodell besitzt.

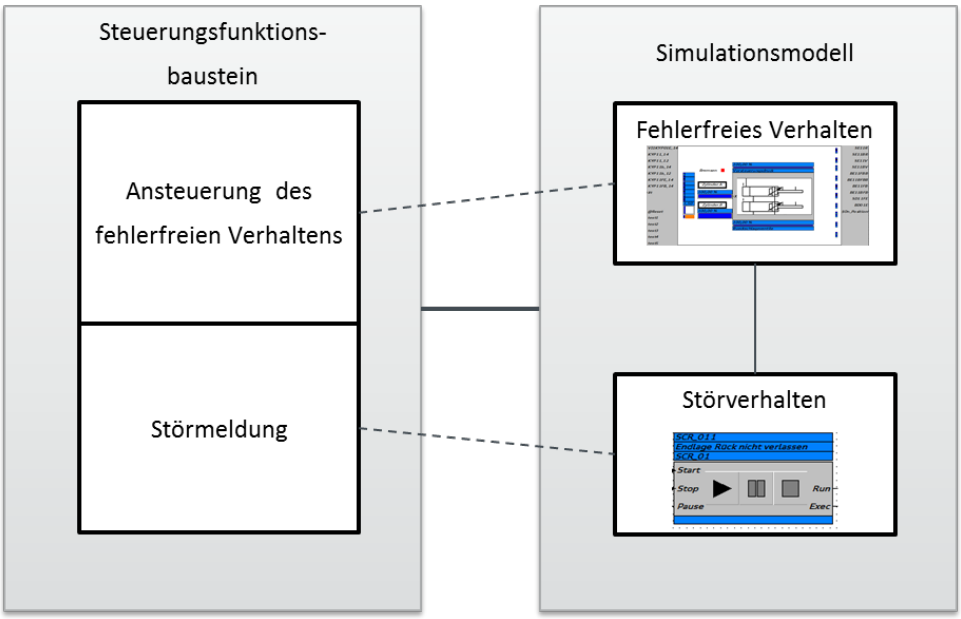


Abbildung 5.13: Erweiterung des Simulationsmodells um das Störverhalten

Es ist deshalb sinnvoll, eine repräsentative Auswahl von Störszenarien durch systematische Störverhaltensmodelle abzubilden, die das von dem normalen Zustand abweichende Verhalten einer Anlage/Komponente beschreiben (vgl. Abbildung 5.13). Die Störverhaltensmodelle sind dann in dem Simulationsmodell zu integrieren. Dadurch werden die Modellausgangssignale generiert, die den Signalen von einer realen Komponente im Störfall entsprechen. Mit diesen Störsignalen ist dann auch die Überprüfung der Steuerungsfunktionalität für das abnormale Verhalten möglich.

Hinzuweisen ist, dass eine systematische Integration des Fehlerverhaltens in die Simulationsmodelle nicht Stand der Technik ist. Dies wird in dieser Dissertation auch nicht weiter behandelt.

5.3.2 Ableitung der Störszenarien aus Fehlersensitive Spezifikation

5.3.2.1 Einschränkungen der herkömmlichen Prozesse

Störszenarien bilden das unerwünschte Systemverhalten dar, was auch Ausgangspunkt von vielen klassischen Sicherheitsanalysemethoden sind. Um die Störszenarien zu finden, werden herkömmlich mit zusätzlichen Prozessen wie z.B. Fehlerbaumanalyse (engl. *fault tree analyse*/ FTA [HRVG81]) analysiert.

FTA verknüpft das unerwünschte Ereignis mit den potentiellen Ursachen durch eine Baumstruktur. Hier startet man bei bekannten Systemversagen und verfolgt den Baum zurück zu möglichen Fehlerursachen. Somit werden mögliche Fehlerursachen sukzessiv identifiziert und anschließend zu dem gesamten Systemmodell hinzugefügt (Abbildung 5.14, oberer Teil). Dieser Prozess wird in [OR04] als „Funktionale Spezifikation“ genannt, bei dem jedoch eine systematische und formale Methode fehlt, was leicht dazu führt, dass ein oder mehrere potentiellen Ursachen übersehen werden. Eine vollständige Bestimmung aller Fehlerursachen ist schon für viele einfachen Systeme nicht trivial. Gerade für viele sicherheitskritischen Systeme, die bestimmte inhärente und eigenständige Funktionen enthalten, lassen sich viele Fehlerursachen nur durch Fachwissen, Erfahrung und Qualifikation der Fach- und Domänenexperte entdecken. Die Vollständigkeit und Qualität der gesamten Modellierungsergebnisse lassen sich schwer beweisen und hängen stark davon ab, wie gut sich die Fachexperte, die die Modelle erstellen, das System auskennen und die Wirkungsketten, die Unfall verursachen können, verstehen.

5.3.2.2 Grundidee der fehlersensitiven Spezifikation

Um das genannte Problem in Abschnitt 5.3.2.1 entgegenzukommen, bietet fehlersensitive Spezifikation eine formale Methode [Ort05] [OR04]. Dieser liegt die Idee zugrunde, anstatt einer additiven Hinzufügung möglicher Fehlerursachen zu einem fehlerfreien System, wird es von einem „chaotischen Systemmodell“ ausgegangen, welches alle möglichen Systemzustände am Anfang beinhaltet. Ein chaotisches Modell einer Simulationskomponente stellt alle möglichen Kombinationen der Signale dar. Im Gegensatz zur funktionalen Spezifikation, betrachtet die fehlersensitive Spezifikation ein System aus komplementärer Sichtweise, indem fehlerhafte Verhalten schrittweise durch Hinzufügung unterschiedlicher Systemregeln, die abgeleitet werden aus den Randbedingungen des technischen Systems,

aus dem chaotischen System eliminiert werden (Abbildung 5.14, unterer Teil). Zusätzlich bietet diese Methode Möglichkeit, Vollständigkeit aller Fehlerquellen formal durch „Verhaltensäquivalenz“ zu beweisen.

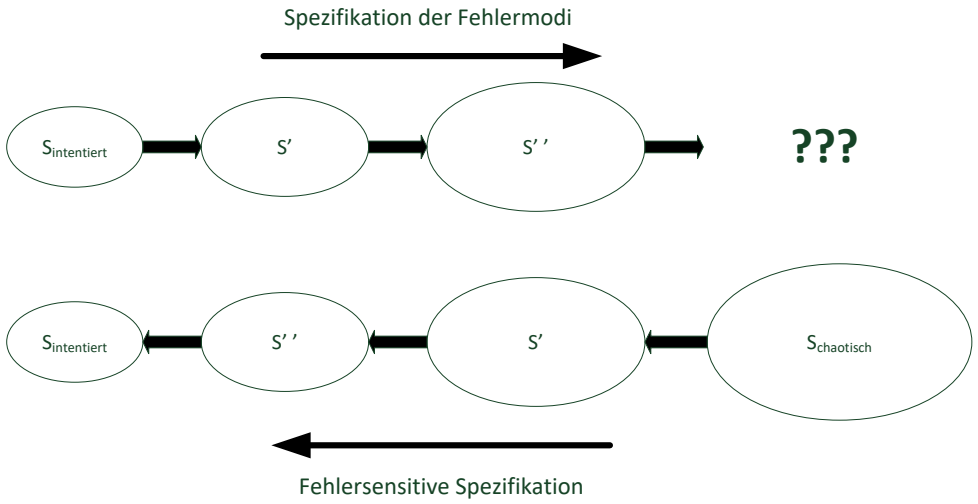


Abbildung 5.14: Standard und Fehlersensitive Spezifikation der Fehlermodi nach [OR04]

5.3.2.3 Vorgehen der Fehlersensitive Spezifikation

Ausgangspunkt der fehlersensitiven Spezifikation ist das zu analysierende System mit definierten Systemgrenzen, die beschreiben, welche Stimuli der Umwelt Einfluss auf das System wirken sowie wie das System darauf reagiert. Das setzt voraus, dass das System sich in Form eines endlichen Zustandsmodells darstellen lässt, was ebenso als funktionales Modell bezeichnet wird. Die Stimuli und Reaktionen werden als boolesche, paarweise disjunkte Signale angenommen.

Bei festgelegten Systemgrenzen lassen sich alle möglichen Kombinationen der Ein- und Ausgangssignale auflisten. Die gesamte Menge der Kombinationen wird als chaotisches Modell dargestellt. Für ein System mit insgesamt n Ein- und Ausgangssignalen sind folglich 2^n mögliche Kombinationen vorhanden. Das chaotische Modell stellt die Vereinigung aller möglichen Szenarien des Systems dar, d.h. sowohl Szenarien für die fehlerfreien Fälle als auch die für die fehlerbehafteten sind im chaotischen Modell enthalten.

Das gewünschte Systemverhalten, das die Systemanforderung spezifiziert, wird durch eine Reihe von Spezifikationsregeln beschrieben. Da jedes fehlerbehaftete Verhalten als solches Verhalten verstanden werden kann, welches dem spezifizierten Verhalten eines Systems widerspricht. Das Fehlerverhalten der Komponente ist ein komplementärer Satz zu Spezifikationsregeln, die zusammen mit dem fehlerfreien Verhalten das chaotische

System aufbauen, in dem alle beliebigen Verhalten enthalten sind. Durch jede eingeführte Spezifikationsregel entsteht demzufolge eine Reihe von Fehlerverhalten bezüglich der Regel, die aus dem chaotischen Modell entfernt werden können.

Dieses Vorgehen wird solange fortgeführt, bis es nur das „gute“ Verhalten am Ende übrig ist. Dadurch werden alle unerwünschten Verhalten schrittweise aus dem chaotischen Modell eliminiert. Gleichzeitig entsteht eine Liste aller mögliche Fehlerszenarien, die sich aus der logischen Negation jeder hinzugefügten Spezifikationsregel ergeben. Schnittmenge aller Spezifikationsregeln wird hier auch als fehlersensitives Modell genannt.

Vollständigkeit der Spezifikation kann durch die „Verhaltensäquivalenz“ nachgewiesen werden, die den Zustand beschreibt, bei dem das fehlersensitive Modell dasselbe Verhalten wie das funktionale Modell aufweist. D.h. die beobachtbare Ein- und Ausgangsverhalten gelten sowohl für funktionales als auch für fehlersensitives Modell.

Eine ausführliche Beschreibung zur fehlersensitiven Spezifikation findet sich in [Ort05]. Dort wird ebenso eine formale Beschreibung und Beweis der Verhaltensäquivalenz gegeben.

5.4 Zwischenfazit

In diesem Kapitel werden einige Probleme in der Verhaltensmodellierung einer Komponente behandelt, die in Abschnitt 3.6 identifiziert wurden. Für verschiedene Anwendungsszenarien sollen passende Modellierungsansätze (Black-Box-, Grey-Box- und White-Box-Modelle) verwendet werden. Generell bietet der Black-Box-Ansatz wenige Detaillierung und Flexibilität jedoch ist der Modellierungsaufwand auch gering. Während ein White-Box-Modell zwar viele Detaillierungen beinhaltet erfordert es aber gleichzeitig einen sehr hohen Aufwand bei der Modellerstellung. Als eine gemischte Variante stellt Grey-Box-Modell ein Kompromiss dar, da die Granularität der berücksichtigten physikalischen Phänomene explizit ausgewählt werden kann. Am Beispiel einer Zylinder-Ventil-Kombination wird aufgezeigt, dass für die meisten Anwendungen in der virtuellen Inbetriebnahmen wie Austaktung und Test der Steuerung Black- und Grey-Box-Modelle gut geeignet sind.

Als gemischte Variante stellt das Grey-Box-Modell einen Kompromiss dar, da die Granularität der berücksichtigten physikalischen Phänomene explizit ausgewählt werden kann.

Abschnitt 5.3 behandelt die Modellierung des Störverhaltens einer Komponente. In industrieller Anlage sind das Störverhalten und die falsche Bedienung in jedem Fall mitzubetrachten, d.h. die abnormalen Zustände sind bei der Modellierung mitzuberücksichtigen. Gerade diese Zustände sollten intensiv getestet werden, um zu erfahren, ob die Anlage/Komponente sicher auf die Störung reagieren kann. Fehlersensitive Spezifikationen nach [Ort05] [OR04]

ermöglichen es, Fehlerfälle systematisch und mit nachweisbarer Vollständigkeit identifizieren zu können. In einem weiteren Teil dieser Dissertation (Abschnitt 8.2.1.3) wird die Ableitung des Störverhalten mit dieser Methode beispielhaft aufgezeigt. Zusätzlich ist zu erwähnen, dass diese Methode solche dynamische Fehlerszenarien behandelt, die ein unerwünschtes Systemverhalten nach Eingabe der binären Signale auslöst. Dieses dynamische Verhalten bestehend aus mit den Systemzuständen sowie Zustandswechseln lässt sich ideal mit dem Zustandsdiagramm beschreiben (vgl.Abschnitt 8.2.1.2). Für statische Fehler eines Systems müssen die statischen Abhängigkeiten der Systemkomponenten (topologisch, parametrisch usw.) berücksichtigt werden. Zusätzliche sind auch quantitative Beziehungen für die Fehlermodellierung zu prüfen. Um diese Anforderungen zu erfüllen, könnte die in Abschnitt 2.2.3 eingeführte Multiple Domain Matrix (MDM) ein möglicher Kandidat sein, indem er ein Hilfsmittel zur statischen Fehlermodellierung und -ableitung. Im Weiteren dieser Dissertation wird diese Aufgabe nicht fortgeführt.

6 Neutrales Austauschformat für die Komponentenmodelle

Aus Sicht eines Anlagenbetreibers hat die Vielzahl an unterschiedlichen mechatronischen Komponenten, die in Fertigungsanlagen verwendet wird, ein unüberschaubares Ausmaß angenommen. Darüber hinaus werden unter dem Stichwort „Industrie 4.0“ weitere Steuerungsfunktionen in die Komponenten integriert, so dass auch die Komplexität der einzelnen Komponenten erhöht wird. Die Anforderungen an Inbetriebnehmer, Instandhalter aber auch Bedienpersonal werden damit weiter erhöht.

Die virtuelle Inbetriebnahme muss diesen erweiterten Anforderungen gerecht werden, soll allerdings den genannten Personengruppen ein Werkzeug zur vereinfachten Arbeitsdurchführung sein. Dies kann kurz wie langfristig nur gelingen, wenn der Modellaufbau einfach und standardisiert erfolgt. Dafür ist eine aktuelle und vollständige Bibliothek von Verhaltensmodellen für alle Komponenten unabdingbar. Als Austauschformat wird in diesem Kapitel dafür eine Erweiterung der Implementierungssprache PLCopen-XML für die Verhaltensbeschreibung vorgeschlagen, die damit einen wichtigen Baustein für eine durchgängige und schnelle Modellerstellung für die virtuelle Inbetriebnahme bildet.

6.1 Ausgangssituation

In Kapitel 4 wird ein neuer Entwicklungsprozess vorgeschlagen, in dem die Entwicklung von Komponenten- und Anlagenmodell getrennt betrachtet wird. Komponentenhersteller sollen für die Entwicklung und Validierung des Komponentenmodells zuständig sein. Für die Wiederverwendbarkeit wird jedes Komponentenmodell in ein projektneutrales Template gespeichert.

Eines der großen Handlungsfelder ist die heterogene Werkzeuglandschaft für die virtuelle Inbetriebnahme, wie z. B. WinMOD [Mew14] von der Fa. Mewes & Partner und SIMIT [Sie09] von der Fa. Siemens, da die Komponentenhersteller für mehrere OEMs in verschiedenen Branchen zuliefern. Jedes VIBN-Werkzeug verfügt über ein proprietäres Dateiformat, das binär codiert ist und daher sich von fremden Werkzeugen nicht importieren oder

exportieren lässt. D.h. Simulationskomponenten, die sich mit einem Werkzeug entwickelt worden sind, können ohne Konvertierung in anderen Werkzeugen nicht verwendet werden. Konvertierung würde jedoch heißen, dass die jeweiligen binären Formate der Werkzeuge offengelegt sind. Komponentenmodelle müssen mehrfach für die verschiedenen Werkzeuge entwickeln und pflegen.

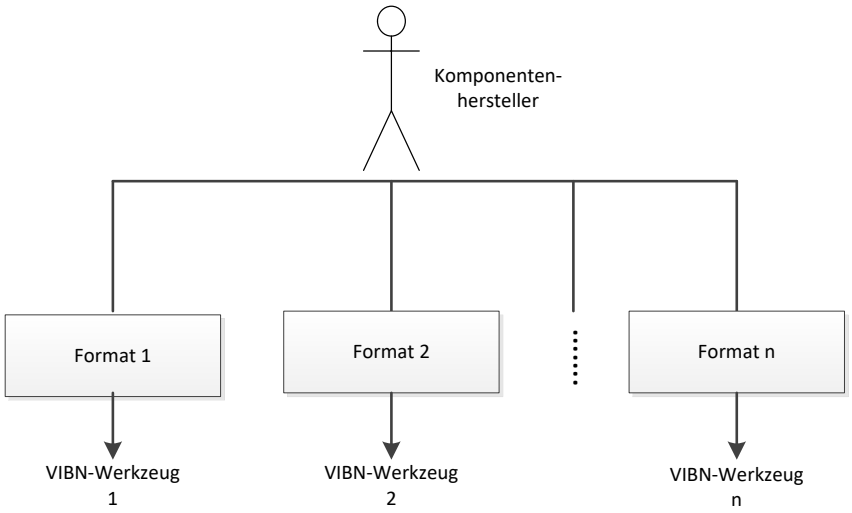


Abbildung 6.1: Komponentenentwicklung für unterschiedliche OEMs

Es ist sehr üblich, dass unterschiedliche OEMs unterschiedliche Werkzeuge gemäß der firmeneigenen Strategie verwenden. Dies bringt wiederum Schwierigkeiten für die Kooperation zwischen OEMs und Komponentenhersteller. Es ist für einen Komponentenhersteller sehr aufwendig, Komponentenmodelle in unterschiedlichen Dateiformaten für verschiedene Kunden als OEMs und Anlagenbauer bereitzustellen. Im ungünstigsten Fall bedeutet es, dass ein Komponentenhersteller jeweils verschiedenartige VIBN-Werkzeuge verwenden muss und für jeden OEM Komponentenmodelle in völlig unterschiedliche Formate bereitstellen soll, wenn jeder OEM sein spezielles VIBN-Werkzeug verwendet (Abbildung 6.1). Das bedeutet für den Komponentenlieferanden höhere Anschaffungskosten für die VIBN-Werkzeuglizenzen, Pflege des Know-hows jedes Zielsystems und größeren Aufwand bei der Erstellung, dem Test und Pflege hinsichtlich Versionierung und Variantenentwicklung der Modellbibliothek.

Ein weiterer Grund der auftretenden Probleme ist die Nutzung von domänenspezifischen Insellösungen in den Planungs- und Implementierungsphasen. Diese führen zu einer suboptimalen Vorbereitung und Integration der Steuerungsinbetriebnahme. Nach der Studie [Tra06] treten die Anlaufstörungen zur 70% aus der Partnerkoordination oder einer mangelhaften Planung bzw. Entwicklung auf (Abbildung 6.2). [Ras90] betont diesen Mangel und stellt fest, dass die meisten Aufwendungen sich auf die Entwicklung von erweiterten aber getrennten Systemen für die Computer Aided Design, Manufacturing, etc. konzentriert haben.

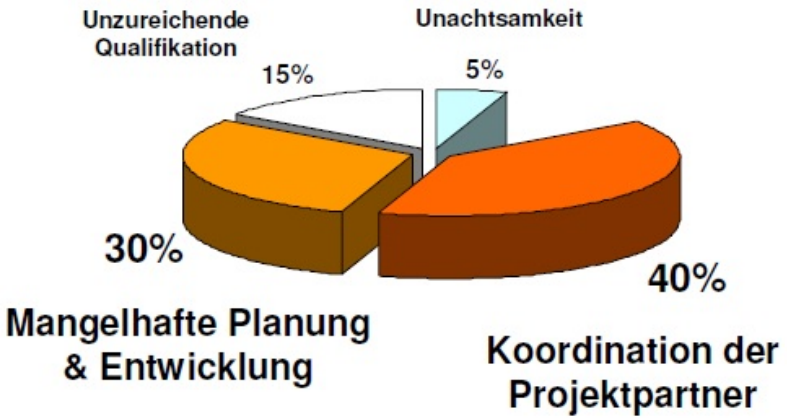


Abbildung 6.2: Fehlerstruktur heutiger Inbetriebnahmen nach [Tra06]

6.2 Lösungsansätze

6.2.1 Proprietäre und neutrale Schnittstelle

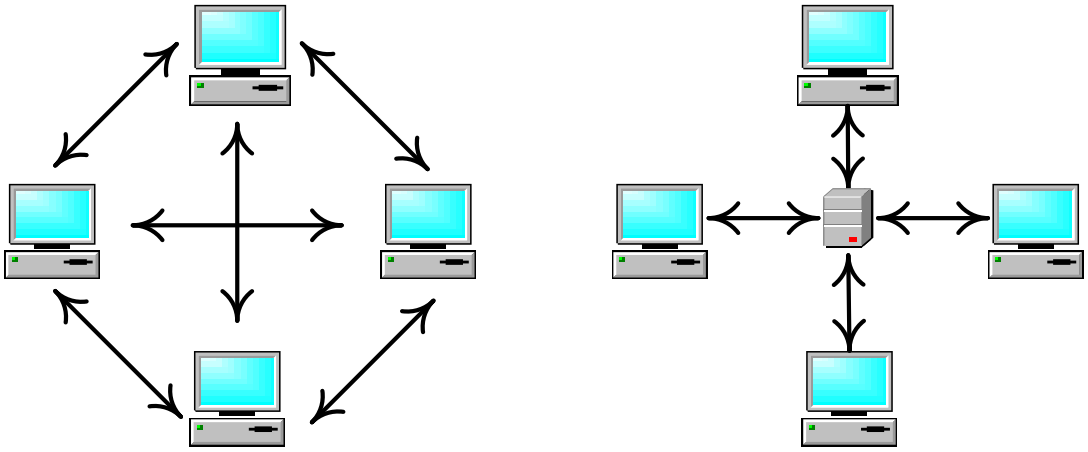
Zur Speicherung von Daten werden meist toolspezifische Datenformate verwendet. Ein direkter Austausch von Daten zwischen verschiedenen Programmen ist daher nicht möglich, sodass der Einsatz von Konvertern notwendig ist, welche die Daten zwischen den einzelnen Formaten umwandeln.

In der Domäne der CAD-Software besitzen viele Softwarewerkzeuge wie CATIA, SolidWorks und NX proprietäre Eigenformate. Allerdings zusätzlich dazu gibt es auch neutrale Formate wie z.B. IGES, STEP und VRML, die ebenso von den etablierten Softwarewerkzeugen unterstützt werden. Dadurch wird die Interoperabilität zwischen unterschiedlichen CAD-Softwarewerkzeugen ermöglicht.

So wie in der CAD-Welt, ist ein standardisiertes neutrales Austauschformat für die Interoperabilität zwischen verschiedenen Simulationswerkzeugen notwendig. Ein neutrales Austauschformat ermöglicht den Austausch der Verhaltensmodelle zwischen VIBN-Werkzeugen und einem effizienten, gesamtwirtschaftlichen Entwicklungsprozess der VIBN-Verhaltensbibliothek.

Generell wird es zwischen proprietären und neutralen Lösungen unterschieden. Proprietäre Lösung sind einem spezifischen Werkzeug zugeordnet. Somit muss zwischen jedem Werkzeugspaar ein neuer und individueller Import/Exportfilter erstellt werden. Für n Applikationen ergibt sich dann $n * (n-1)/2$ Schnittstellen (Abbildung 6.3(a)). D.h. die Anzahl der proprietären Schnittstelle ist exponentiell zur Anzahl der Applikationen.

Dagegen ist eine neutrale Lösung eine offene Lösung, die keine Werkzeugsbindung aufweist. Wird ein offenes Format für den Informationsaustausch zwischen verschiedenen Applika-



(a) $n * (n-1)/2$ proprietäre Schnittstellen

(b) n neutrale Schnittstellen

Abbildung 6.3: Anzahl benötigter Schnittstellen: proprietäre gegen neutrale Lösung

tionen verwendet, so muss für jede Applikation nur eine Schnittstelle erstellt werden. Für n Applikationen werden nur n Schnittstellen benötigt (Abbildung 6.3(b)).

6.2.2 Lösungsmöglichkeiten

Es ist zu sehen, dass die Anzahl der proprietären Direktschnittstellen für den Informationsaustausch zwischen unterschiedlichen Applikationen deutlich größer als die Anzahl der benötigten Schnittstellen bei einem Neutralformat ist. Aus diesem Grund wird im Rahmen dieser Dissertation eine neutrale Lösung vorgeschlagen, die den Datenaustausch zwischen unterschiedlichen VIBN-Werkzeugen ermöglicht. Im Folgenden wird eine Übersicht über einige vorhandene Ansätze gegeben.

Functional Mock-up Interface

Eine detaillierte Beschreibung zu Functional Mock-Up Interface (FMI) sowie Functional Mock-up Unit (FMU) wurde bereits in Abschnitt 3.1.4 gegeben. Zwar bietet FMI eine Möglichkeit, dass unterschiedliche Simulationssoftware unter einem gemeinsamen Framework neutral zusammenarbeiten [Mod]. Leider besitzen die meisten aktuell etablierten VIBN-Werkzeuge bisher noch nicht diese Technologie [Mew14] [Sie09] [Des].

Agentenbasierte Co-Simulation

Die in Abschnitt 3.1.5 eingeführte multiagentenbasierte Struktur ermöglicht auch eine Co-Simulation von Modellen, die in unterschiedlichen Simulationswerkzeugen entwickelt werden. Die Agentenstruktur (siehe Abbildung 3.1) ermöglichte es, dass die zunächst proprietären Simulationsmodelle über die Agentenschnittstellen miteinander kommunizieren können. Dafür muss ein traditionelles Simulationswerkzeug um eine Schnittstelle erweitert werden, die folgende drei Funktionen erfüllen [JSW18]:

- Kommunikation zwischen Modellen,
- prozessorientierte physische Interaktion,
- Synchronisation für die Simulation.

Jede Schnittstelle besteht aus einen generischen und einen spezifischen Teil. Der generische Teil ist in Verbindung mit dem Agenten und bleibt unverändert, da der verbundene Agent sich auch immer gleich verhält. Der spezifische Teil der Schnittstelle koordiniert die Simulationswerkzeuge und muss nach dem Simulationswerkzeugtyp separat entwickelt werden. Es ist auch möglich, Hardwarekomponenten mit einem Agenten nach der Struktur wie Abbildung 3.1 auszustatten. Die daraus entstehende HiL-Schnittstelle kann dann in die agentenbasierte Co-Simulation eingebunden werden [JJK+20]. Für die Implementierung der prozessorientierten Interaktion an der Schnittstelle sollte der angeforderten Detaillierungsgrad der zu simulierenden Modelle berücksichtigt werden.

AutomationML

Ein Konsortium, bestehend u.a. aus den Firmen ABB, Daimler, KUKA, Siemens, Volkswagen hat einen Standard, der ein Datenformat für die Speicherung und den Austausch von Anlagenplanungsdaten namens AutomationML [Aut16] beschreibt, entwickelt. AutomationML wird in der IEC 62714 standardisiert. Dieser offene Standard spezifiziert ein Datenaustauschformat, das den durchgängigen Datenaustausch im Engineering ermöglicht. AutomationML ist keine eigenständige Sprache sondern setzt aus verschiedenen vorhandenen Standards bzw. Datenformaten zusammen, die über stark typisierte Links verbunden sind (vgl. Abbildung 6.4). Dafür werden verschiedene Dateiformate für die unterschiedlichen Bereiche wie Topologie, Geometrie, Kinematik und Logik benutzt, die alle auf dem XML-Format basieren [Dra09].

Zur Darstellung der Struktur- und Topologieinformationen wird die Systembeschreibungssprache CAEX verwendet, die als Dachformat für AutomationML darstellt. Objektstruktur lässt sich durch CAEX beschreiben. Verschiedene Aspekte eines Objekts werden diesen Objekten direkt zugeordnet. Darüber hinaus können Objekte Eigenschaften und Schnittstellen zu anderen Objekten besitzen.

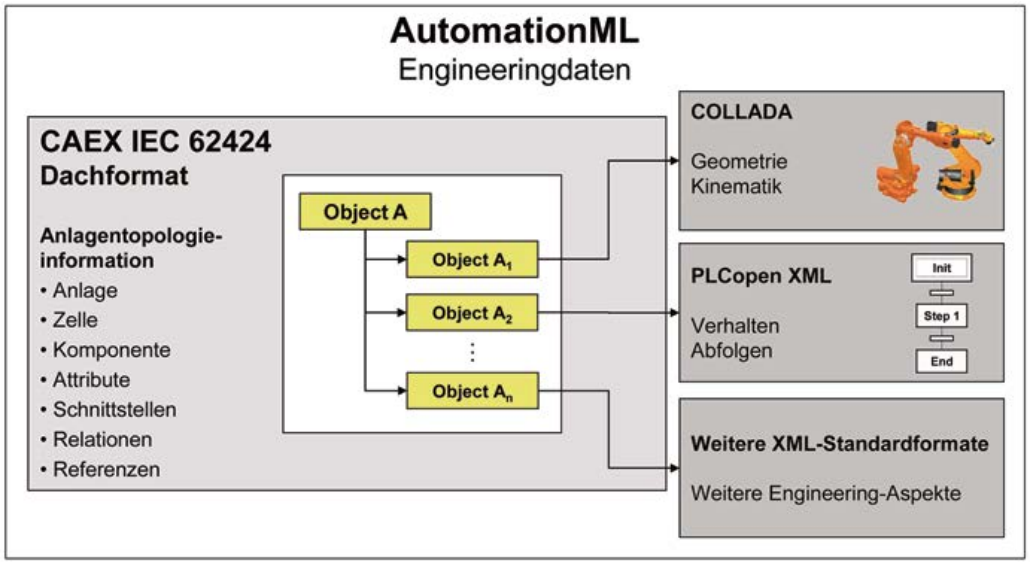


Abbildung 6.4: Architektur von AutomationML [Dra09]

Ursprünglich wird das Collaborative Design Activity (Collada) Datenaustauschformat für 3D-Grafiken und Animationen im Bereich von Computerspielen entwickelt, welches ein XML-basiertes Datenaustauschformat mit der spezifischen Dateiendung $*.dae$ ist. Innerhalb von AutomationML wird Collada eingesetzt, um Geometrien und Kinematiken von Objekten zu beschreiben. Unter Geometrie wird dabei das äußere Erscheinungsbild (im allg. eine 3D-CAD-Zeichung) verstanden. Die Kinematik beschreibt die Bewegungsfreiheitsgrade eines Objektes, d.h. an welchen Punkten sich Achsen (Gelenke) befinden.

Durch die Architektur, in der CAEX als Dachformat ausgewählt und COLLADA, PLCopen XML sowie weitere XML-Standartformate miteinbezogen werden, ist AutomationML ein offener und erweiterbaren Standard, der den durchgängigen Datenaustausch im Engineering ermöglicht.

IEC 61131-3 und PLCopen XML

IEC 61131 wurde von der International Electrotechnical Commission genormt, welches ein herstellerunabhängiger Standard für speicherprogrammierbare Steuerung darstellt. IEC 61131 befasst sich mit den Anforderungen der SPS-Hardware sowie des Programmiersystems. Die darin enthaltene IEC 61131-3 [IEC13] ist eine Richtlinie zur SPS-Programmierung, in der sowohl die Syntax als auch die Semantik von insgesamt fünf verschiedenen Programmiersprachen festgelegt, die wieder in drei Gruppen unterteilen lassen:

- Textuelle Programmiersprachen: Anweisungsliste (AWL), Strukturierter Text (ST)

- Graphische Programmiersprachen: Kontaktplan (KOP), Funktionsbausteinsprache (FBS)
- Strukturelle Sprache: Ablaufsprache (AS)

Als eine der fünf in IEC 61131 genormten Programmiersprachen ist FBS eine datenflussorientierte Sprache mit gerichtetem Graph, bei der die Ein- und Ausgänge von Funktionsbausteinen miteinander verbindet werden, um Systemverhalten zu beschreiben. Wichtige Sprachelemente gemäß IEC 61131 sind Funktionsbausteine und Funktionen, die als „Blackbox“ mit Eingangs- und Ausgangsparametern dargestellt werden: z.B. $\geq$ (oder), & (und), SR-Flipflop, TON (Zeitstufen), CTUD (Zähler), mathematische Funktionen.

Um die Steuerungsprogramme gemäß IEC 61131-3 zwischen verschiedenen Programmierumgebungen auszutauschen, wird PLCopen XML entwickelt, das eine Abbildung der Steuerlogik gemäß IEC 61131-3 in einem herstellerunabhängigen Format ermöglicht [PLC08]. PLCopen XML ist XML-Schema-basiert und unterstützt die Speicherung und Weitergabe der fünf normierten Sprachen in IEC 61131-3. Dadurch wird beschaffen, Steuerungsprogramme zwischen unterschiedlichen Steuerungswerkzeugen auszutauschen. Bei der Darstellung mittels PLCopen XML werden nicht nur textuelle Informationen sondern auch grafisches Layout gespeichert, wie z.B. Position eines Objekts der grafischen Sprachen oder Verbindungslinien zwischen Objekten. Somit können eben die graphischen Programmiersprachen wie KOP und FBS ohne Informationsverlust zwischen verschiedenen Werkzeugen übertragen werden.

6.3 Neutrales Austauschformat basiert auf PLCopen XML

6.3.1 Erweiterung der IEC 61131-3-Bibliothek

Es kann festgestellt werden, dass viele gängigen VIBN-Werkzeuge (WinMOD, SIMIT) einen funktionsbausteinbasierten Ansatz verwenden. Dafür steht in jedem Werkzeug eine vorgefertigte Standard-Bibliothek mit Funktionsbausteinen zur Verfügung, die logische, mathematische, regelungs- und messtechnische Funktionen erhält. Aus diesen Grundbausteinen werden dann die Komponentenmodelle für die Simulation erstellt.

Dieses Bibliothekskonzept ähnelt sich dem FBS-Ansatz, der ebenso in IEC 61131-3 standardisiert ist. FBS handelt sich um eine grafische Programmiersprache, in dem die Ein- und Ausgänge der Funktionsbausteine miteinander verschaltet werden. Diese Netzwerkbeschreibung aus Funktionsbausteinen und Verbindungslinien lässt sich durch Instanziierung ständig von der Steuerung aufrufen. Eine Bibliothek mit einer Reihe von booleschen

und arithmetischen Funktionen und Funktionsbausteinen sowie einigen Zeitgliedern wird in IEC 61131-3 standardisiert. Mit vielen vorhandenen Standardelementen, z.B. UND, ODER, GT, LT etc., lassen sich schon viele einfachen Komponenten darstellen. Für die komplizierteren Verhalten wird der Modellierungsumfang von Standard FBS-Bibliothek nach IEC 61131 überschritten, da die prinzipiell ursprünglich für die Programmierung von Steuerungsprogrammen und nicht für die Verhaltensmodellierung ausgelegt sind.

In dieser Dissertation wird vorgeschlagen, die Standardbibliothek von IEC 61131-3 um Funktionen und Funktionsbausteine für die Verhaltensmodelle von Anlagenkomponenten zu erweitern. Umfang der Erweiterung ist abhängig von den Anwendungsbereichen der zu modellierenden Komponenten. Analysen von Simulationsmodellen haben gezeigt, dass der hier beschriebene FB-Bibliotheksumfang die typischen Komponentenverhalten in umfassender Weise beschreiben. Für die VIBN-Modellierung biete es sich an, dass die erweiterte Bibliothek den gleichen Funktionsbausteinumfang aufweist wie die Bibliotheken der VIBN-Werkzeuge.

Da in dieser Dissertation WinMOD als das Implementierungswerkzeug für die VIBN-Modellierung ausgewählt wird, was ein Repräsentant einer Werkzeugklasse mit dem funktionsbausteinbasierten Ansatz darstellt, wird ein Vergleich von dem Bibliotheksumfang von WinMOD gegenüber IEC 61131-3 durchgeführt. Das Vergleichsergebnis wird in Tabelle A.1 in Anhang A aufgelistet.

Es ist zu erkennen, dass die in IEC 61131-3 standardisierten Funktionsbausteine eine Teilmenge darstellt, im Vergleich zu allen Bausteinen, die in WinMOD-Bibliothek enthalten sind. Damit der Funktionsumfang der Komponentenmodelle mit der IEC 61131-3 FB-Bibliothek erstellt werden kann, muss die Standardbibliothek von IEC 61131 Funktionsbausteinen entsprechend dem Umfang der VIBN-Werkzeuge erweitert werden.

Abbildung 6.5 zeigt auszugsweise ein Beispiel, wie der Funktionsbaustein „I-Glied“ erstellt wird. Die innere Darstellung des Funktionsbausteins gliedert sich in zwei Teile: Deklarationsteil (nicht in der Abbildung dargestellt) und Implementationsteil. Im Deklarationsteil werden die Datenstrukturen, in diesem Beispiel die sieben Eingänge sowie ein Ausgang deklariert. Der Implementationsteil bildet den Rumpf eines Funktionsbausteins, in dem die Algorithmen mit der SPS-Programmiersprache Strukturierten Text (ST) beschrieben werden. Dadurch werden die wesentlichen Funktionalitäten wie hier im Beispiel vom von „I-Glied“ als ein Funktionsbaustein vollständig beschrieben. Die Deklaration sowie der Inhalt der Algorithmen lassen sich in einen Funktionsbaustein kapseln und anschließend als Typ-Definition in einer Bibliothek abspeichern.

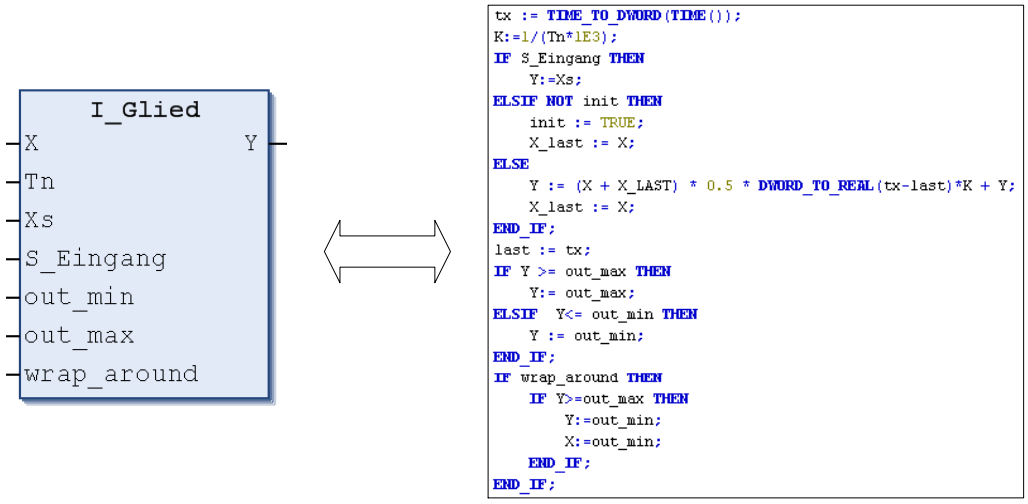


Abbildung 6.5: Erweiterung von IEC 61131-3 um ein Bibliothekselement „I-Glied“

6.3.2 Exportierung in PLCopen XML

Sowohl der Inhalt mit Deklaration und Algortihmus als auch die eingekapselte E/A-Beschaltung eines Funktionsbausteins lassen sich mit PLCopen XML darstellen.

In [PLC08] wird festgelegt, dass die textuelle Programmiersprachen Anweisungsliste und Strukturierter Text (Abschnitt 6.2.2) als formatierter XHTML-Text dargestellt werden müssen. XHTML (Extensible Hypertext Markup Language) ist eine XML-gerechte Formulierung Serialisierung der Hypertextsprache HTML [P$^+$00]. Auf Grund der gleichen syntaktischen Basis wie XML ermöglicht XHTML auch den Datenaustausch zwischen unterschiedlichen Werkzeugen, die kein Browser sind. In Anhang B wird die komplette XML-Notation des Funktionsbausteins „I-Glied“ als Beispiel für die Bibliothekserweiterung gezeigt.

In PLCopen XML wird jeder Funktionsbaustein als eine Program Organization Unit (POU) betrachtet. Der eigentliche Funktionsbaustein wird durch das Element <block> modelliert, in dem im Attribute *typenName* der Name des Funktionsbausteins zugewiesen wird. Zusätzlich dazu werden in <block> die Ein- und Ausgänge der POU definiert. Die Eingangs- und Ausgangsvariablen werden durch die Elemente <inVariable> und <outVariable> dargestellt. Ein- und Ausgänge einer POU lassen sich entweder mit einer lokalen Variablen oder mit den Ein- und Ausgängen einer anderen POU verbinden. Das erstellte PLCopen XML Dokument ist im Anhang B gezeigt.

6.4 Angestrebter Entwicklungsprozess

Abbildung 6.6 stellt den modifizierten Entwicklungsprozess der Verhaltensmodellentwicklung dar, der aus folgenden fünf Schritten besteht:

1) Voraussetzung ist eine wie oben beschrieben, erweiterte IEC 61131-3 Bibliothek. Jeder neue Funktionsbaustein dieser erweiterten Bibliothek muss gut validiert und getestet werden, damit er industriell tauglich ist. Der Umfang der Bibliothek muss so erweitert werden, dass er die Anforderungen der VIBN-Modellierung erfüllt.

2) Komponentenhersteller entwickelt mittels der erweiterten IEC-61131-3 Funktionsbausteinbibliothek das Komponentenmodell und testen ihre Komponentenverhaltensmodelle, was bisher Aufgabe der OEMs ist.

3) Das entwickelte Komponentenmodell wird in PLCopen XML Format exportiert

4) Das in PLCopen XML beschriebene Komponentenmodell wird an die Kunden (OEMs, Anlagenbauer usw.) zusammen mit der realen Komponente geliefert

5) Das Komponentenmodell wird wiederum vom Kunden in das jeweilige VIBN-Simulationswerkzeug importiert. In diesem werden die unterschiedlichen Komponentenverhaltensmodelle zu einem Anlagenmodell zusammengebaut. Voraussetzung ist, dass das VIBN-Werkzeug eine Importschnittstelle für PLCopen XML bietet.

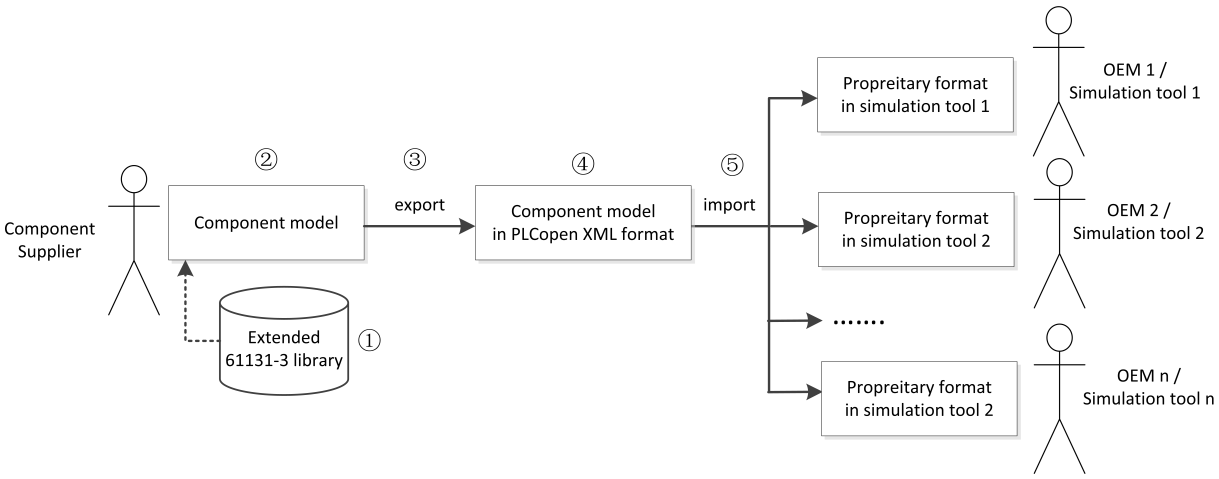


Abbildung 6.6: Modifizierter Entwicklungsprozess

6.5 Zwischenfazit

Damit die Komponentenmodelle zwischen den Modellentwicklern und den Simulationswerkzeugen ausgetauscht werden können, wird in diesem Kapitel zunächst eine erweiterte

Bibliothek basiert auf der Standardbibliothek von IEC 61131-3 Funktionsbausteinen vorgestellt. Diese erweiterte Bibliothek ist wiederum über das PLCopenXML Format neutral austauschbar. Das von PLCopen entwickelte XML Format ermöglicht die Wiederverwendung der Komponentenmodelle durch Import- und Exportfunktion zwischen unterschiedlichen VIBN-Werkzeugen. Dieses kann auch in AutomationML eingebunden werden.

Die Vorteile bestehen darin, dass die Komponentenhersteller die Komponentenmodelle VIBN-Werkzeugneutral erstellen und pflegen können. Dadurch wird die heute übliche Praxis, dass die OEM die Komponentenmodelle selbst entwickeln abgelöst.

Es ist zudem zu erwähnen, dass das in diesem Kapitel eingeführte neutrale Austauschformat eine praktische Lösungsmöglichkeit für den Austausch der Komponentenmodelle bietet. Anlagenmodelle, die in einem proprietären Format vorhanden sind, können vom Simulationsmodellentwickler in das neutrale Austauschformat übersetzt werden. Dies ermöglicht es, diese Simulationskomponenten mittels Import-Funktion in die gewünschte Simulationsumgebung zu übernehmen. Es bestehen jedoch auch andere Lösungsansätze wie z.B. Co-Simulation mit FMI oder Multiagentensystem, in denen Komponentenmodelle mit verschiedenen Formaten unter demselben Framework zusammenarbeiten. Die Komponentenmodelle z.B. im multiagentenbasierten Ansatz können in einer traditionellen Simulationsumgebung entwickelt werden, die jedoch um ein Agenteninterface erweitert werden muss. Dieser Prozess gehört nicht zum Schwerpunkt der vorliegenden Dissertation. Dieser Entwicklungsprozess ist eindeutig dem unterlagerten V-Modell für den Entwicklungsprozess der virtuellen Komponenten (Abschnitt 4.3.3.1) zuzuordnen. Die Abbildung des normalen und fehlerbehafteten Verhaltens gilt methodisch wie in Kapitel 5 ausführt. Die Möglichkeiten zur Integration der Komponentenmodelle in ein Anlagenmodell wird in diesem Kapitel diskutiert und nicht weiter als Schwerpunkt behandelt.

7 Testframework

Um die hohen Kosten bei der Vielzahl miteinander zusammenarbeitender mechatronischer Komponenten bei auftretenden Fehlern während der realen Inbetriebnahme zu minimieren, sollen möglichst alle Komponenten, Teilsysteme und das gesamte Automatisierungssystem während der virtuellen Inbetriebnahme realitätsnah getestet werden. Die funktionalen Verhalten der einzelnen Komponenten inklusiv die Zusammenarbeit mit den Steuerungsfunktionsbausteinen müssen vorher getestet werden. Um den Testaufwand zu minimieren wäre eine automatisierte Durchführung des Tests „auf Knopfdruck“ gewünscht.

In diesem Kapitel wird die Systemstruktur und -funktionen eines Testframeworks beschrieben, das entwickelt wird, um die Durchführung des Tests in der virtuellen Inbetriebnahme zu automatisieren und optimieren.

7.1 Einführung

Generelle gliedert sich eine Testumgebung in das Testsystem und System Under Test (SUT) (Abbildung 7.1).

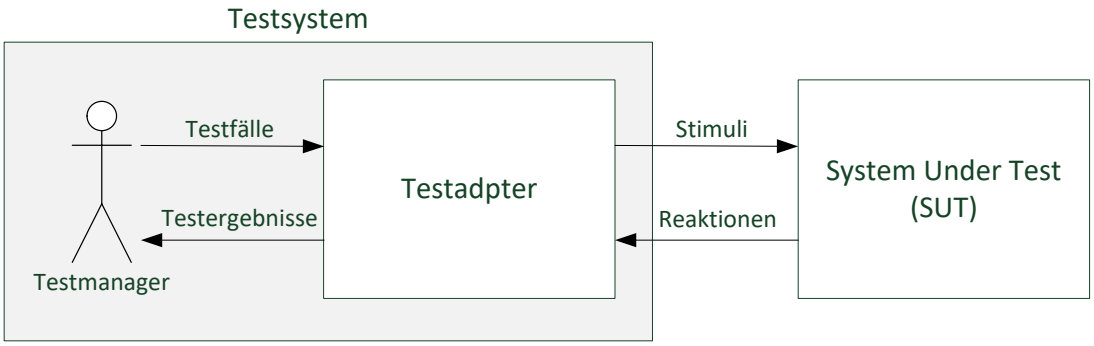


Abbildung 7.1: Generelle Struktur einer Testumgebung

Das Testsystem hat neben dem Management der Testsuiten die Aufgaben, die Testfälle abzuarbeiten, die Testergebnisse aufzunehmen und zu bewerten. Es besteht aus zwei Komponenten:

- Testmanager, der die Testfälle erstellt und an das Testsystem weitergibt. Gleichzeitig werden die Testergebnisse von dem Testmanager empfangen und evaluiert. Die Korrektheit der Systemfunktionen in Bezug auf die Anforderung wird hier überprüft und dokumentiert.
- Testadapter, der einerseits die Testfälle in die ausführbaren Testfälle konvertiert und die Stimulisignale auf SUT ausübt, andererseits empfängt der Testadapter die Reaktionen von SUT und leitet die Testergebnisse weiter an Testmanager.

SUT stellt das zu testende Objekt dar, das auf die Stimuli reagiert und die Ausgangswerte weiter an das Testsystem liefert. Zu SUT gehören alle implementierten Teile, die durch den Test validiert werden sollen. Wie in Abschnitt 4.3 beschrieben, unterliegt die Implementierung des SUTs bestimmten Anforderungen. Die Erfüllung der einzelnen Anforderungen wird durch Testfälle überprüft.

Ein Testfall ist der ausführbare zeitliche Ablauf aus mehreren Testschritten, der die Funktionalität eines Systems überprüft. Im Wesentlichen besteht ein Testfall aus drei Teile (Abbildung 7.2):

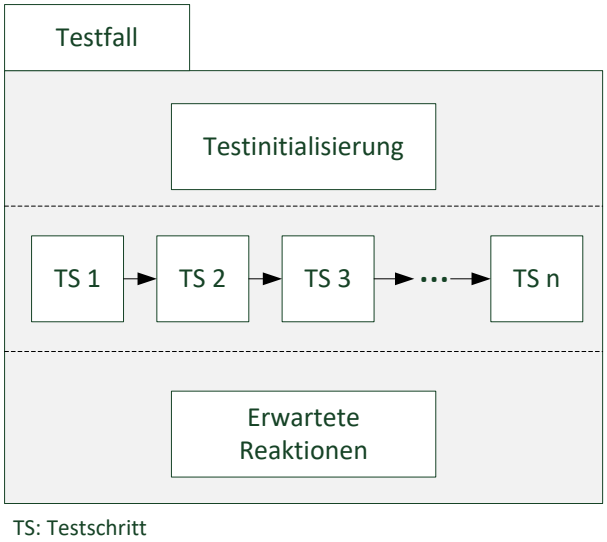


Abbildung 7.2: Bestandteile eines Testfalls

Testinitialisierung: Jeder Testfall setzt einen gewissen Initialzustand voraus, damit der Testfall durchführbar ist. Weicht der Anfangszustand eines Testobjekts von dem vorgesehenen Ausgangszustand ab, muss das Testobjekt zunächst initialisiert werden. Nur bei der richtigen Testinitialisierung werden weitere Testschritte ausgeführt.

Testschritt: Jeder Testfall besteht aus einer Reihe von Testschritten, die den Hauptteil eines Testfalls darstellt. Testschritt ist das Grundelement eines Testfalls und lässt sich nicht mehr teilen. Normalerweise besteht ein Testfall aus mehreren Testschritten, die zeitlich miteinander verbunden werden. Jeder Testschritt stellt einen operativen Schritt

dar, der entweder das zu testende Objekt stimuliert mit bestimmten Eingabewerten, oder die Reaktionen des Testobjekts mit definierten Vorgaben einlesen.

Erwartete Reaktionen: Erwarteten Reaktionen beschreiben die Ergebnisse, wenn das Testobjekt korrekt auf die Stimuli reagiert, die durch die Systemanforderung definiert werden. Testergebnisse werden mit den erwarteten Reaktionen verglichen. Bei Übereinstimmung kann der Test als „erfolgreich“ markiert werden.

7.2 Handlungsbedarf

Stand der Technik ist, dass ein großer Anteil von Tests in der virtuellen Inbetriebnahme manuell ausgeführt wird. D.h. die Aufgaben von „Testmanager“ und „Testsystem“ in Abbildung 7.2 von einer Person (Tester) übernommen werden, der die einzelnen Testfällen ausführt und die Testergebnisse manuell analysiert und bewertet.

Es wurde schon in Abschnitt 2.3 geschrieben, dass der Aufwand zur Erstellung eines Simulationsmodells gemäß der Anforderungsspezifikation nicht zu unterschätzen ist. Einen wesentlichen Bestandteil der Modellierung stellt der Testprozess dar, indem die einzelnen Testfälle definiert und durchgeführt werden. Mit dem im letzten Abschnitt vorgestellten Testsystem lassen sich die Testfälle zwar semi-automatisiert ausführen, jedoch ist der Gesamtaufwand nicht zu vernachlässigen.

Es bestehen einige Handlungsfelder, um die Durchführung des Tests weiter zu automatisieren und den Aufwand zu verringern.

- Die manuelle Vorgehensweise der Testdurchführung ist wegen der großen Anzahl von Testfällen zeitaufwendig. Unter Berücksichtigung der Tatsache, dass die Komponentenmodellierung ebenso aufwandintensiv ist (vgl. Abschnitt 2.3), steht einem produktiven Einsatz der virtuelle-Inbetriebnahme-Technologie der großer Gesamtaufwand gegenüber.

- Während des Projektablaufs ist es nicht vermeidbar, dass ein Komponentenmodell durch die Versionierung ständig aktualisiert wird. Bei solcher Änderung müssen dann die betroffenen Testfälle manuell neu ausgeführt werden.

- Das SUT hat eine verteilte Struktur, in der die beiden Testobjekte (Steuerung und Simulationswerkzeug) über eine Kommunikationsschnittstelle verbunden werden. Da der Test auf die Zusammenarbeit von Komponentenmodell und Steuerungsfunktionsbaustein erzielt ist, müssen die Stimuli gleichzeitig auf die beiden Objekte ausgeübt werden. Für die manuelle Bedienung bedeutet es, dass der Tester gleichzeitig Signale

von den beiden Testobjekten setzen oder zurücksetzen soll, was praktisch manchmal nicht bedienbar ist.

- Es wird nicht nur das Normalverhalten sondern auch das Störverhalten eines Komponentenmodells sowie die Fehlermeldungsfunktionen von Steuerungsfunktionsbaustein getestet (siehe auch Abschnitt 5.3). Um Fehler in das Komponentenmodell zu injizieren, setzt es voraus, dass der Tester Kenntnisse sowohl über die Bedienung des Simulaitonswerkzeugs als auch die Verhalten und die Modellierung der Komponente besitzt. Heute wird die virtuelle Inbetriebnahme meistens von einem Steuerungstechniker durchgeführt, der nicht zwangsläufig viele Erfahrungen mit der Simulation besitzt. Eine zusätzliche Qualifikation ist erforderlich. Dies erhöht die Einstiegshürde in die virtuelle Inbetriebnahme.

Auf der Grundlage der obengenannten Handlungsfeldern wird in dieser Arbeit ein Testframework zur Verbesserung der Testdurchführung in der virtuellen Inbetriebnahme vorgeschlagen. Das Testframework stellt die notwendigen Infrastrukturen bereit für den automatisierten Test der Komponentenmodelle und Steuerungsfunktionsbaustein in der virtuelle Inbetriebnahme und bietet folgende Eigenschaften an:

Automatisierung der Testdurchführung: Der manuelle Bediener in Abbildung 7.1 als Testmanager wird um einen „Testassistent“ erweitert, der in der Lage ist, Testfälle auszulesen und anschließend automatisch auszuführen. Nach einer Änderung des SUTs können die Tests mit keinem oder geringem Aufwand erneuert wiederholt werden.

Fehlerinjektion: Es wird in Abschnitt 5.3 über unterschiedliche Technologien zur Fehlerinjektion diskutiert. Da die Fehler direkt in das Komponentenmodell nachgebildet werden müssen, wird ein Fehlerinjektor als ein Aufsatz zum Simulationswerkzeug implementiert. Für ein Fehlerszenario werden bestimmte zugängliche Stellen des Komponentenmodells manipuliert, um die gezielten Fehlerzustände zu generieren. Ähnlich wie Testfall besteht eine Fehlinjektion für ein bestimmtes Fehlszenario aus mehreren Schritten, die zeitlich nacheinander abgelaufen sind. Über ein tabellarisches Interface lassen sich die Fehlinjektionen eingeben. Die Eingabe setzte keine tieferen Kenntnisse über die Bedienung des Simulationswerkzeugs voraus.

Testbewertung: Ein ausgeführter Testfall kann nur bewertet werden, wenn das erwartete Ergebnis bekannt ist, was in einen Testfall festgelegt wird (vgl. Abschnitt 7.1). Das Soll- und Ist-Verhalten wird durch Testarbiter verglichen, der dann ein „Verdict“ bestimmt, der das Testergebnis beinhaltet und eine der folgenden Möglichkeiten aufweisen:

- **Erfolgreich**: Das Testergebnis entspricht den Erwartungen. Der Testlauf ist erfolgreich.

- **Fehlgeschlagen**: Das Testergebnis entspricht nicht den Erwartungen. Der Testlauf ist nicht erfolgreich.
- **Fehler**: Während der Testdurchführung ist ein Fehler aufgetreten. Grund dafür können z.B. Fehler in der Testumgebung oder fehlerhafte Testfälle sein.

Ein Modell gilt als validiert, wenn alle Testverdicts als „Erfolgreich“ bewertet werden.

7.3 Testframework

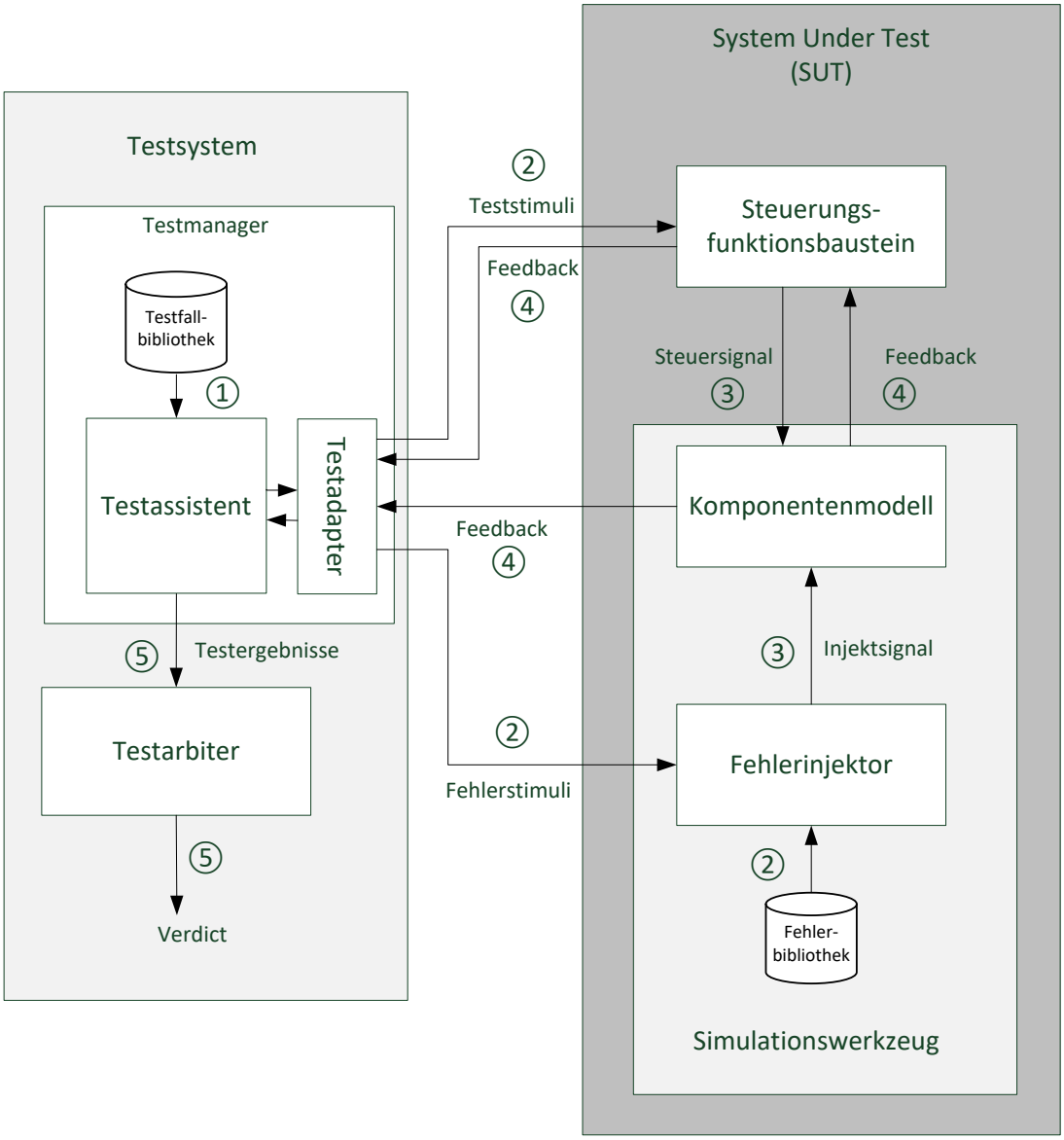


Abbildung 7.3: Testfrmework

Die Testdurchführung im Rahmen des Frameworks bestehen aus folgenden wesentlichen Schritten:

1. Auslesen der Testfälle: Alle Testfälle werden im Vorfeld entsprechend der Systemspezifikation erstellt und in einer Testfallbibliothek gespeichert. Bei jeder Testfalldurchführung wird ein Testfall aus der Bibliothek ausgelesen und in den Testassistent geladen.

2. Nachdem der Testassistent die initialen Bedingungen eines Testfalls geprüft und gegebenenfalls eingestellt hat, führt er den Testfall in einzelnen Testschritt aus, indem die Teststimuli jeweils an den Steuerungsfunktionsbaustein und den Fehlerinjektor ausgeübt werden. Entsprechend dem Fehlerstimulisignal wird das richtige Fehlerszenario aus der Fehlerbibliothek ausgelesen und in den Fehlerinjektor geladen.

3. Das Steuersignal wird von Steuerungsfunktionsbaustein auf das Komponentenmodell übertragen. Um das gewünschte Fehlerverhalten in das Komponentenmodell einzubringen, wird ein Signal von Fehlerinjektor auf das Komponentenmodell injiziert. Falls es sich um den Test des Normalverhaltens handelt, wird kein Fehler injiziert.

4. Das Komponentenmodell reagiert auf beide Signale und gibt Feedbacks jeweils zurück an den Steuerungsfunktionsbaustein und Testmanager. Das Feedback an den Steuerungsfunktionsbaustein wird gleich auch an den Testmanager weitergeleitet.

5. Feedbacks werden von Testmanager zusammengefasst und an den Testarbiter weitergegeben, der das Testergebnis interpretiert und als Verdict bestimmt.

7.4 Zwischenfazit

In diesem Kapitel werden zunächst Anforderungen an das angestrebte Testsystem definiert. Insbesondere werden einige Handlungsfelder, die die Vorgehensweise der Testdurchführung optimieren können, weiter betrachtet. Im Anschluss wird die Struktur eines konzeptionellen Testframeworks vorgestellt. Zusätzlich zu einem klassischen Testsystem mit Testadapter, Testassistent, Testarbiter etc., verfügt das Testframework über einen Teil, der in das SUT integriert ist. Dieser Teil hat die Aufgabe, Fehlerstimuli durch die Injektsignale in die Komponenten einzubringen und Rückmeldung wieder an das Testsystem zu übertragen. Wesentliche Schritte der Testdurchführung mit dem Testframework werden erläutert. In Abschnitt 8.2.2 wird das konzeptionelle Testframework in Form eines prototypischen Testwerkzeugs umgesetzt.

8 Fallstudien

In diesem Kapitel werden die Konzepte, die in vorherigen Kapiteln entwickelt wurden, durch zwei Fallstudien umgesetzt. Die erste Fallstudie zeigt, wie die Komponentenmodelle mithilfe des erweiterten Austauschformats, das auf dem IEC 61131-3 Standard basiert, unternehmenübergreifend ausgetauscht werden können. Mit der zweiten Fallstudie soll erläutert werden, wie die normalen und fehlerhaften Verhalten einer VEP-Lineareinheit mit einem prototypischen Werkzeug getestet werden. Inhalt dieses Kapitels basiert zum Teil auf der Vorveröffentlichungen des Autors in [LBD+15] und [LDGSD15]

8.1 Fallstudie 1: Neutrales Austauschformat

Im Folgenden wird der Entwicklungsprozess des Komponentenmodells am Beispiel eines SEW-Frequenzumrichters mit dem Applikationsmodul „Erweiterte Buspositionierung“ erläutert. Anschließend wird das entsprechende Komponentenmodell mittels des im letzten Kapitel konzeptionell vorgestellten, neutralen Formats darstellt.

8.1.1 Modellierung eines SEW-Frequenzumrichters

Ein Frequenzumrichter ist eine typische antriebstechnische Komponente für die fördertechnischen Anwendungen in der Automobilindustrie. Der SEW-Frequenzumrichter mit dem Applikationsmodul „Erweiterte Buspositionierung“ eignet sich besonders für Anwendungen, bei denen die Positionen durch Vorgabe der Solldrehzahl und eventuell einer Beschleunigungsrampe gesteuert werden. Zur Verhaltensmodellierung kann auf die Herstellerdokumentation [SEW10b] [SEW10a] sowie die Eplan-Daten [Gis15] zurückgegriffen werden.

Im ersten Schnitt soll die Schnittstelle zur Steuerung für das Verhaltensmodell definiert werden. Das generelle Verhalten des Moduls wird durch 12 Prozessdatenworte bestimmt (Tabelle 8.1). Die 6 Prozesseingangsdatenworte am Modellausgang liefern zusätzlich zu einem Statuswort aktuelle Position und Geschwindigkeit an die SPS. Das Modellverhalten wird durch 6 Prozessausgangsdatenworte am Modelleingang bestimmt, die unter anderen Informationen über Freigabe, Betriebsmodus sowie aktuelle Position, Geschwindigkeit

PA	Prozessausgangsdaten	PE	Prozesseingangsdaten
PA1	Steuerwort 2 (2 Byte)	PE1	Statuswort
PA2	Zielposition High	PE2	Ist-Position High
PA3	Zielposition Low	PE3	Ist-Position Low
PA4	Soll - Geschwindigkeit	PE4	Ist-Geschwindigkeit
PA5	Beschleunigungsrampe	PE5	Wirkstrom
PA6	Verzögerungsrampe	PE6	Geräteauslastung

Tabelle 8.1: Ein- und Ausgangsprozessdatenworte von SEW IPOSs

und Ausgangsströme enthalten. Die restlichen Ein- und Ausgänge haben keinen Einfluss auf das generelle Verhalten des Modells. Sie werden entweder direkt von Eingangs- auf Ausgangsseite weitergeleitet oder mit einem Wert vorbelegt.

Nach der Schnittstellendefinition wird das Modellverhalten abgebildet. Als Grundlage dienen hier die Ablaufdiagramme aus dem Handbuch des Applikationsmoduls [SEW10a]. Das Verhaltensmodell setzt sich aus zwei Teilen zusammen. Die Zustandssteuerung stellt die Sollwerte bereit für die drei unterstützten Betriebsarten: Tipp-Betrieb, Referenzier-Mode und Automatikbetrieb. Den zweiten Teil bildet der Antrieb selbst, aus dem die Ist-Position und -Geschwindigkeit berechnet werden. Der zeitliche Verlauf der Ist-Geschwindigkeit ist abhängig vom vorgegebenen Sollwert und der Rampenzeit. Abbildung 8.1 veranschaulicht die Oberfläche des erstellten Verhaltensmodells.

8.1.2 Darstellung des Frequenzumrichtermodells mit dem vorgeschlagenen neutralen Format

Das in letztem Abschnitt entwickelte Frequenzumrichtermodell wird mit dem in Kapitel 3 vorgestellten IEC 61131-3 Konzept in der Steuerungsentwicklungsumgebung CoDeSys [3S-10] dargestellt. Eine große Anzahl von Funktionen im WinMOD-Modell (z.B. UND, ODER, GT, LT etc.) kann direkt mit dem entsprechenden IEC 61131-3 Bibliothekselementen ersetzt werden. Damit das SEW-Antriebsmodell vollständig dargestellt werden kann, werden entsprechend der vorgestellten Erweiterung die fehlenden Funktionsbausteine wie „I-Glied“, „Grenzwertschalter“, „Analog-Digital-Konverter“, „Digital-Analog-Konverter“, „Binär-Digital-Konverter“ und „Digital- Binär-Konverter“, die nicht in der IEC-Bibliothek vorhanden sind, in CoDeSys implementiert. Sie werden anschließend als benutzerdefinierte Bibliothek gespeichert und verwendet. Somit ist eine 1:1 Abbildung aller Funktionsbausteine des Frequenzumrichtermodells von WinMOD auf CoDeSys nun möglich. Die Identität wird im nächsten Abschnitt validiert, in dem gezeigt wird, dass die Modelle mit der erweiterten IEC 61131-3 Bibliothek die gleichen Verhalten aufweisen können, wie die ursprüngliche Implementierung mit WinMOD. Abbildung 8.2 zeigt einen Ausschnitt von dem erstellten

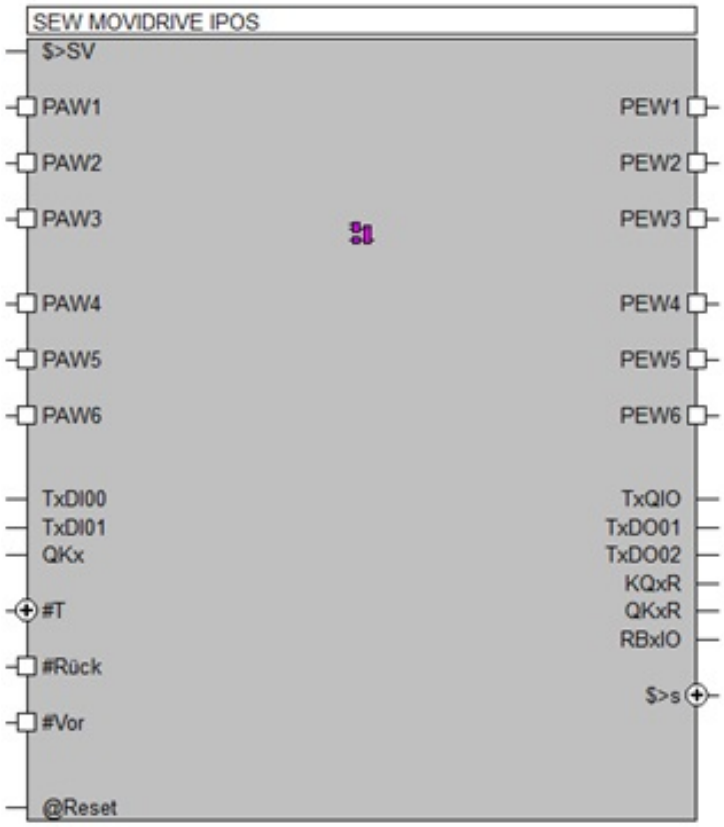


Abbildung 8.1: Modell eines SEW-Antriebes in Form des WinMOD-Makros

Modell in CoDeSys.

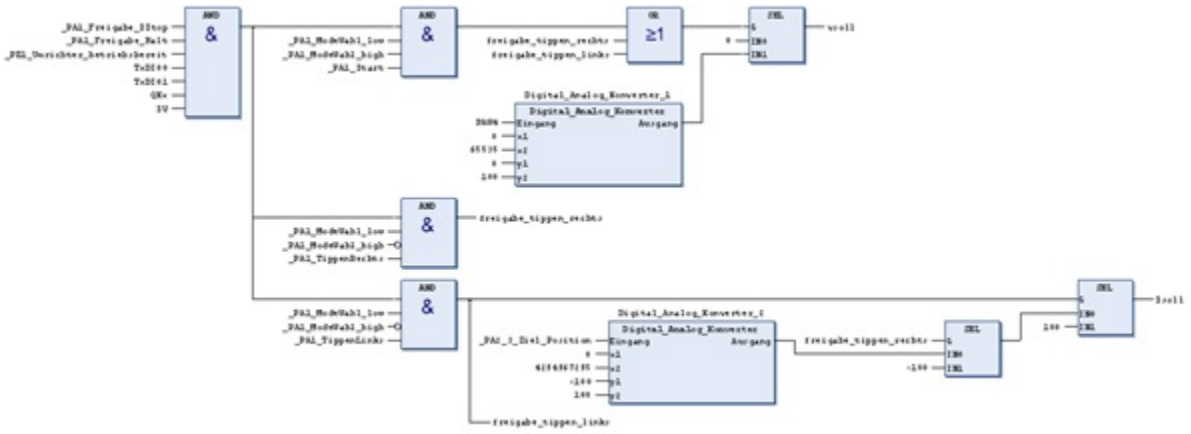


Abbildung 8.2: Umsetzung des internen Aufbaus eines WinMOD-Modells mit der erweiterten IEC 61131-3 Bibliothek

Die parametrierten und verdrahteten Funktionsbausteine können dann in einen neuen Funktionsbaustein (Abbildung 8.3) zusammengefasst werden, der die gleichen Ein- und Ausgänge, wie das WinMOD-Modell enthält und auch das gleiche Verhalten aufweisen soll. In CoDeSys besteht darüber hinaus die Möglichkeit, das erstellte Modell in PLCopenXML Format zu exportieren, das wiederum in ein anderes Werkzeug, das Importfunktion von

PLCopenXML-Datei unterstützt (z.B. SIMIT), importieren lässt. Somit ist das erstellte Antriebsmodell über das neutrale Format PLCopenXML austauschbar.

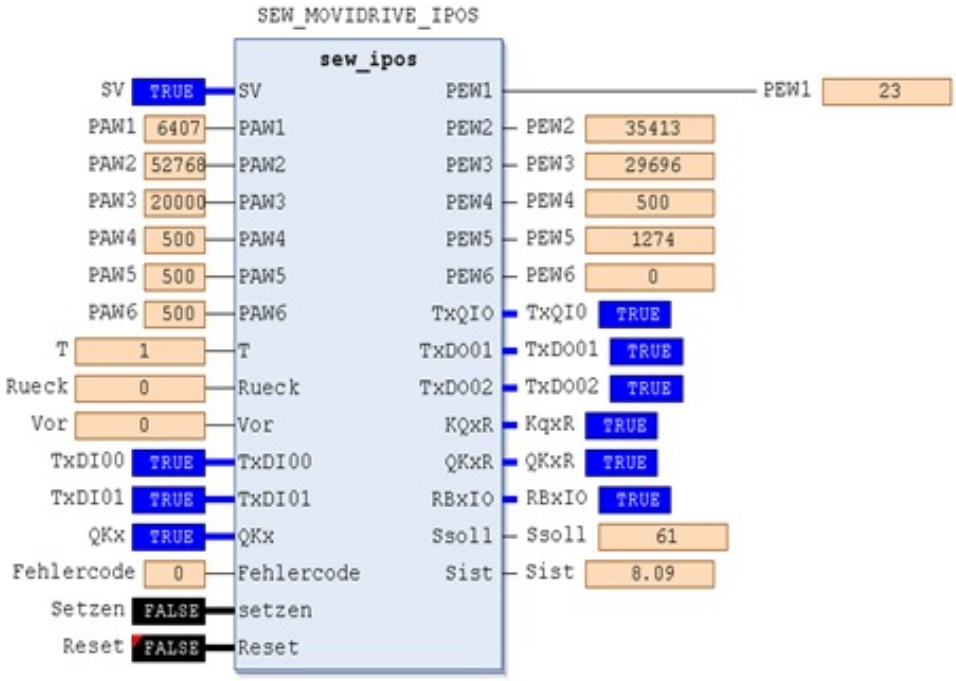


Abbildung 8.3: Modell eines SEW-Antriebes in Form eines Bausteines nach IEC61131-3

8.1.3 Validierung des erstellten Modells

Im Handbuch [SEW10a] werden Taktdiagramme angegeben, die das Zeitverhalten von Frequenzumrichtern für unterschiedliche Betriebsmodi beschreiben. Abbildung 8.4 zeigt ein Beispiel für den Automatikbetrieb.

Um das in WinMOD erstellte Simulationsmodell und das mit der erweiterten IEC 61131-3 Bibliothek erstellte Modell validieren zu können, muss deren zeitliches Verhalten mit den Verhalten in Taktdiagrammen, die hier als Spezifikation gelten, verglichen werden.

Sowohl in WinMOD als auch in CoDeSys können Signale während der Simulation gezielt auf einen bestimmten Wert gesetzt werden, um das im Taktdiagramm dargestellte Verhalten wiederzugeben. Im Automatikbetrieb können die Zielposition (PA2 und PA3), die Soll-Geschwindigkeit (PA4), die Beschleunigungsrampe (PA5) und die Verzögerungsrampe (PA6) über die Prozessausgangs-Datenworte vorgegeben werden. Die Positionierfahrt startet mit Bit PA1:8 „Start“ = „1“. Wenn der Antrieb die Zielposition erreicht hat, wird im Statuswort PE1 das Bit PE1:3 „Ziel-Position erreicht“ auf „1“ gesetzt. Wird beim gesetzten PA1:8 = „1“ eine neue Zielposition vorgegeben, fährt der Antrieb sofort diese neue Position an (vgl. auch [SEW10a]). Die aktuellen Werte von Position und Geschwindigkeit sowie das Sensorsignal „Ziel-Position erreicht“ werden aufgezeichnet und

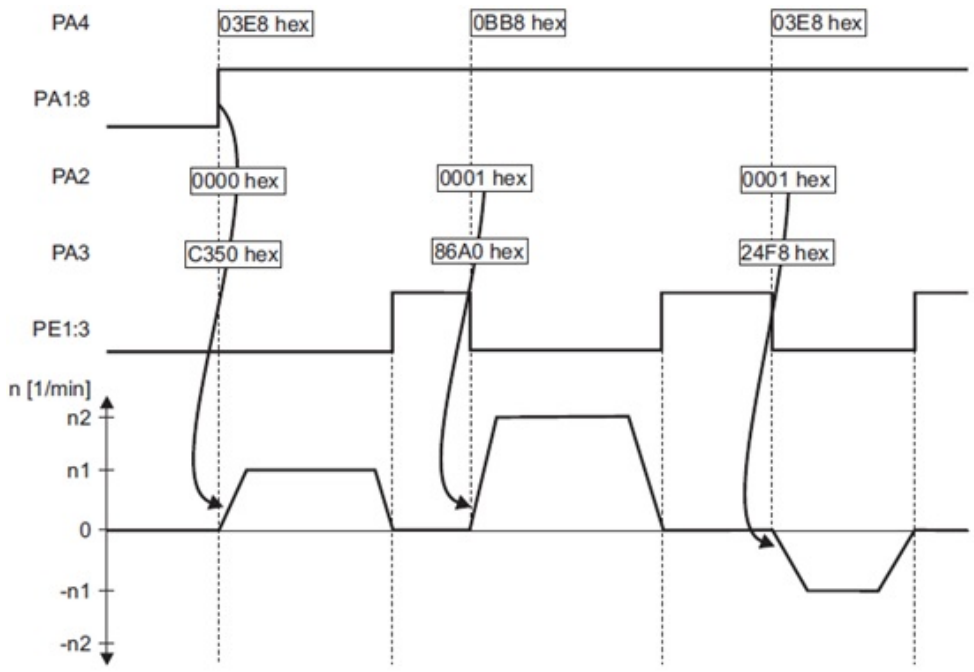


Abbildung 8.4: Taktdiagramme eines SEW-Antriebes im Automatikbetrieb

für das Simulationsmodell in WinMOD (Abbildung 8.5) sowie das mit der erweiterten IEC 61131-3 Bibliothek dargestellte Modell (Abbildung 8.6) jeweils in einem Diagramm dargestellt.

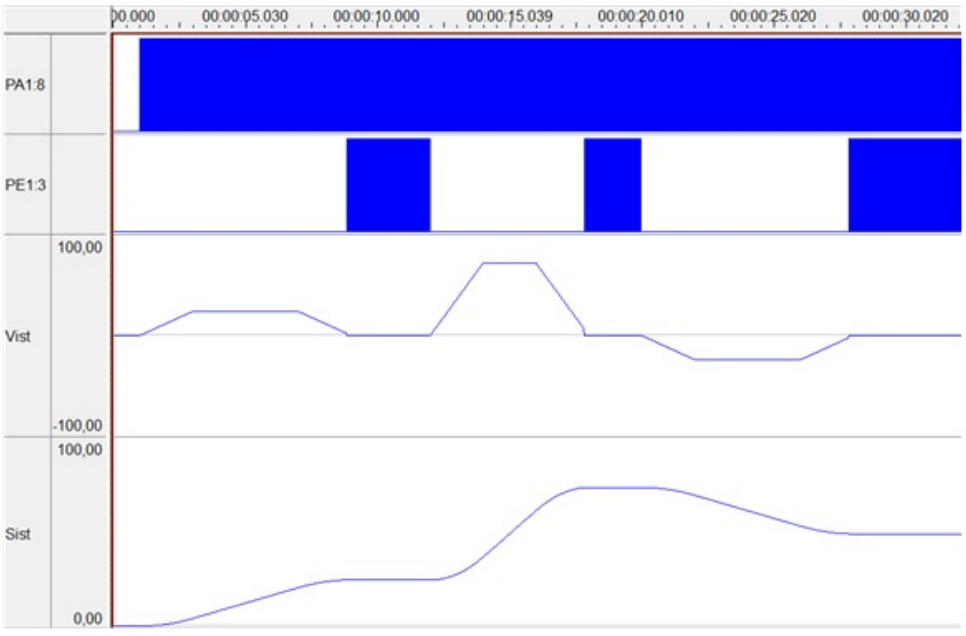


Abbildung 8.5: Zeitliche Signalverläufe des WinMOD-Modells im Automatikbetrieb

Es ist zu sehen, dass die Signale in den beiden Abbildungen gleiche zeitliche Verläufe, wie die in Abbildung 8.4 aufweisen. Dadurch werden die beiden Verhaltensmodelle validiert.

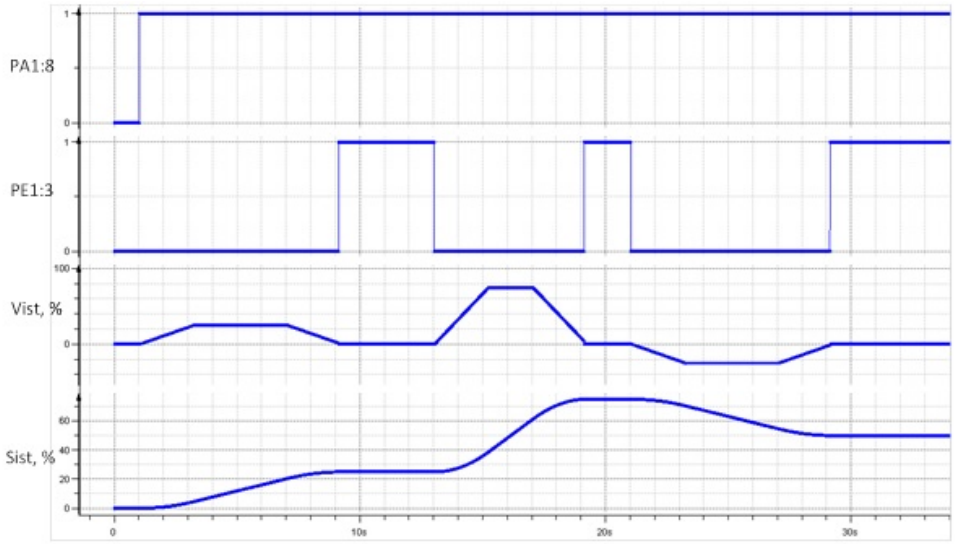


Abbildung 8.6: Zeitliche Verläufe von Signalen des erstellten Modells im Automatikbetrieb

8.1.4 Zwischenfazit

In diesem Abschnitt wird der Modellierungsprozess einer Komponente am Beispiel eines SEW-Frequenzumrichters mit dem Applikationsmodul „Erweiterte Buspositionierung“ aufgezeigt. Darüber hinaus wird ein neutrales Austauschformat verwendet, welches den Austausch und die Wiederverwendung der Komponentenmodelle ermöglicht. Anschließend wird das Frequenzumrichtermodell mit dem neutralen Austauschformat dargestellt, verglichen und evaluiert, dass das Modell das Verhalten richtig wiedergibt.

8.2 Fallstudie 2: Testen der normalen und fehlerbehafteten Verhalten

8.2.1 Abbildung der Störverhalten einer VEP-Lineareinheit

Im Folgenden wird aufgezeigt, wie das Verhaltensmodell einer pneumatisch getriebenen Lineareinheit der Fa. „VEP“ abgebildet werden kann. Bei der Modellierung wird sowohl das fehlerfreie Verhalten als auch das Störverhalten, welches in Abschnitt 5.3.1 eingeführt wird, berücksichtigt.

8.2.1.1 Abbildung des Normalverhaltens in WinMOD

Die zu modellierende Lineareinheit eignet sich für die Handhabungs- und Positionieraufgabe. Sie ist mit einem oder zwei doppelwirkenden 5/2-Wegventilen (je nach Größe der

Lineareinheit) und doppelwirkenden Antriebszylindern ausgestattet, der in der „Vor“ und „Rück“ Stellung verriegelt werden kann. Die Druckluftanschlüsse des Zylinders werden über ein vorgeschaltetes Wegventil beaufschlagt, die den Schlitten zwischen verstellbaren Anschlägen verfahren. Diese Funktionsgruppe hat als Eingangssignale die von der SPS an das Ventil gehenden Signale und als Ausgangssignale die Sensorsignale des Zylinders, die wiederum zurück an die SPS gehen.

Das Verhaltensmodell wird mit der Simulationsumgebung WinMOD erstellt. Als erster Schritt für die Verhaltensmodellierung soll die Schnittstelle zur Steuerung definiert werden. In Tabelle 8.2 und Tabelle 8.3 werden jeweils die Ein- und Ausgangssignale des WinMOD-Makros „Zylinder-Ventil-Kombination“ aufgelistet:

Name	Type	Kommentar
Frei_V	binär	Wenn „Frei_V“=1, soll Zylinder nach vorne gefahren werde
Frei_R	binär	Wenn „Frei_R“=1, soll Zylinder zurück gefahren werden
Ver_V	binär	Wenn „Ver_V“=1, ist Zylinder in Vorlage entriegelt
Ver_R	binär	Wenn „Ver_R“=1, ist Zylinder in Rücklage entriegelt
Timer_Vor	binär	„Timer_Vor“=1, wenn die Zeit die Zylinder zum Fahren von der Rücklage zur Vorlage benötigt, abläuft
Timer_Rück	binär	„Timer_Rück“=1, wenn die Zeit die Zylinder zum Fahren von der Vorlage zur Rücklage benötigt, abläuft
#t	analog	Parametrierbar zwischen 0 bis 10 Sekunden
@Reset	binär	Zurücksetzen der Komponente
testcase	digital	Auswahl der Testcase

Tabelle 8.2: Eingangssignale (Steuerungsausgänge) des WinMOD-Makros „VEP-Lineareinheit“

Name	Type	Kommentar
Rück	binär	Wenn „Rück“=1, ist Zylinder eingefahren
Vor	binär	Wenn „Vor“=1, ist Zylinder ausgefahren
$On_Position	analog	Position von Zylinder

Tabelle 8.3: Ausgangssignale (Steuerungseingänge) des WinMOD-Makros „VEP-Lineareinheit“

Das Analogeingangssignal „#t“ dient zur Parametrierung der zeitlichen Dauer des Vor- und Rücklaufs des Ventils. Die beiden Signale „@Reset“ und „testcase“ sind für die Abbildung des Störverhaltens reserviert (siehe Abschnitt 8.2.1.2). Die restlichen Signale werden direkt über PROFINET mit dem SPS-Ausgang angeschlossen.

Bei der Ausgangsseite des Modells liefert das Signal „$On_Positio“ aktuelle Position an SPS zur Darstellung in 3D-Modell.

Nach der Schnittstellendefinition wird das Modellverhalten abgebildet, was durch das Zeitablaufdiagramm spezifiziert wird. Im Zeitablaufdiagramm werden die zeitlichen Abläufe der charakteristischen Ein- und Ausgangssignale der Lineareinheit aus einzelnen Schritten beschrieben.

Da für dieses Anwendungsszenario ausreichend ist, wenn das Modell über die Ein- und Ausgangssignale mit SPS kommunizieren kann, wird für die Modellierungstiefe „Blackbox-Modellierung Ebene 2“ ausgewählt. Die Pneumatik, die sich im Inneren der Funktionsgruppe befindet, muss bei der Modellierung daher nicht explizit beachtet werden.

Für die Modellierung des Verhaltens dieser Funktionsgruppe müssen nun abhängig von den Ansteuerungssignalen des Ventils die jeweiligen Sensorsignale generiert werden. Da für einen ausreichenden Detaillierungsgrad keine physikalische Berechnung für die Bewegung des Zylinders abhängig von der einströmenden Druckluft benötigt wird, können einfach zwei Parameter festgelegt werden, welche die Zeit angeben, die der Zylinder für eine Vorwärts- bzw. Rückwärtsbewegung benötigt.

Die Signalübertragung zwischen Ein- und Ausgängen wird in WinMOD signalflussorientiert durch einzelne Funktionsblocks dargestellt, die wiederum in WinMOD in einem sogenannten WinMOD-Makro zusammengefasst werden können.

8.2.1.2 Ableitung des Störverhaltens

Die Störverhalten der VEP-Lineareinheit soll nach der in Abschnitt 5.3.1 vorgestellte Methode „fehlersensitive Spezifikation“ abgeleitet werden.

Festlegung der Systemgrenze

Als Ausgangspunkt steht die Wahl der Systemgrenzen, die alle Ein- und Ausgaben des Systems beinhaltet. Als Eingaben gelten alle Signale, die einen Einfluss auf das Verhalten des Systems haben können. Ausgaben beschreiben alle von System produzierten Aktionen, die gleichzeitig auch beobachtbar sind. Tabelle 8.4 fasst alle Zustandsübergänge zusammen.

Um die Systemgrenzen zu identifizieren ist hier ein Zustandsdiagramm hilfreich (vgl. Abbildung 8.7).

Da die beiden Endlagen der Lineareinheit mechanisch verriegelt werden können, um bei einem eventuellen Druckabfall zu sichern, müssen die beiden verriegelten Zustände in dem Zustandsdiagramm ebenso berücksichtigt werden.

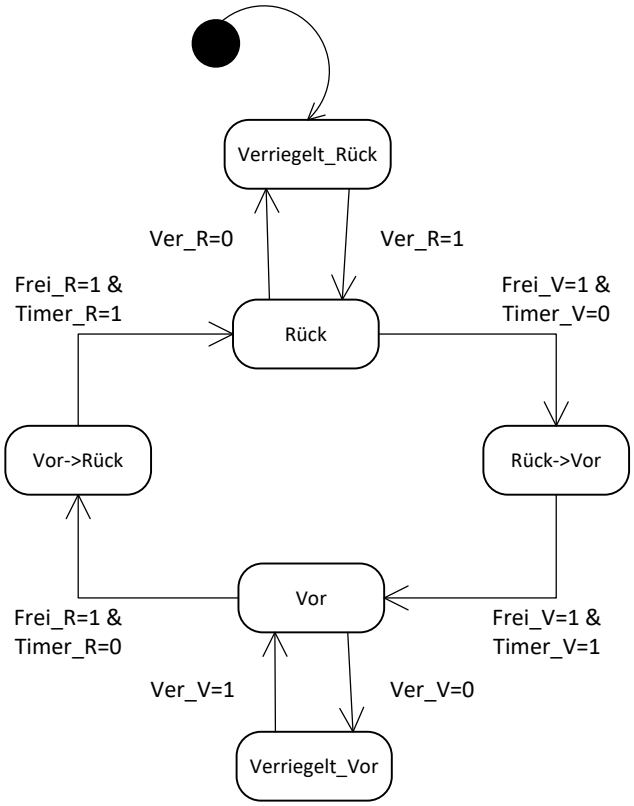


Abbildung 8.7: Zustandsdiagramm der VEP-Lineareinheit

- **Verriegelt_Rück**: Der Zylinder ist in der Endlage „Rück“ und verriegelt
- **Rück**: Der Zylinder ist in der Endlage „Rück“ und entriegelt
- **Rück→Vor**: Der Zylinder befindet sich auf dem Übergang von der Endlage „Rück“ nach der Endlage „Vor“
- **Verriegelt_Vor**: Der Zylinder ist in der Endlage „Vor“ und verriegelt
- **Vor**: Der Zylinder ist in der Endlage „Vor“ und entriegelt
- **Vor→Rück**: Der Zylinder befindet sich auf dem Übergang von der Endlage „Vor“ nach der Endlage „Rück“

Die Lineareinheit befindet sich solange in der Endlage „Verriegelt_Vor“ bis die beiden Signale „Ver_V“ und „Frei_V“ auf 1 gesetzt wird. Dabei wird die Lineareinheit durch das Steuersignal „Ver_V“ entriegelt. Da es keine Rückmeldung zu diesem Zustandswechsel gibt, ist der Zustandswechsel von „Verriegelt_Vor“ nach „Vor“ von außen nicht beobachtbar. Aus dem Grund wird im Systemgrenzmodell nur die beobachtbare Ausgabe „Vor“ berücksichtigt. Der Zustandsübergang von „Vor→Rück“ nach „Rück“ wird durch das gesetzte Signal „Timer_R=1“ bestimmt. Dasselbe gilt auch für die Zustandsübergänge von der Endlagenposition „Rück“ nach „Rück→Vor“ und anschließend „Vor“ (vgl. Abbildung 8.7).

Zustandsübergang	Aktion
Verriegelt_Rück ⇒ Rück	Ver_R=1
Rück ⇒ Verriegelt_Rück	Ver_R=0
Verriegelt_Vor ⇒ Vor	Ver_V=1
Vor ⇒ Verriegelt_Vor	Ver_V=0
Rück ⇒ Rück→Vor	Frei_V=1 & Timer_V=0
Rück→Vor ⇒ Vor	Frei_V=1 & Timer_V=1
Vor⇒ Vor→Rück	Frei_R=1 & Timer_R=0
Vor→Rück ⇒ Rück	Frei_R=1 & Timer_R=1

Tabelle 8.4: Übergänge des Zustandsdiagramms

Insgesamt bestehen die zu betrachtenden Systemgrenzen aus zwölf Eingabe- und vier Ausgabesignalen:

- Eingaben: $\Gamma_{in} = \{Verriegelt\_R\ddot{u}ck_{in},\ Verriegelt\_Vor_{in},\ Ver\_R_{in},\ Ver\_V_{in},$
  $R\ddot{u}ck_{in}, Vor_{in},\ R\ddot{u}ck \rightarrow Vor_{in},\ Vor \rightarrow R\ddot{u}ck_{in},\ Frei\_R_{in},$
  $Frei\_V_{in},\ Timer\_Vor_{in},\ Timer\_R\ddot{u}ck_{in}\}$
- Ausgaben: $\Gamma_{out} = \{R\ddot{u}ck_{out},\ Vor_{out},\ R\ddot{u}ck \rightarrow Vor_{out},\ Vor \rightarrow R\ddot{u}ck_{out}\}$

Abbildung 8.8 veranschaulicht das oben beschriebene Systemgrenzenmodell nochmals.

Abbildung des chaotischen Modells

Eine bestimmte Kombination von Ein- und Ausgangssignalen wird als ein Szenario bezeichnet, das ein bestimmtes Systemverhalten bedeutet. Die Gesamtheit aller möglichen Szenarien stellen die Potenzmengen der jeweiligen Menge von Ein- und Ausgaben dar. Diese lassen sich als $\mathcal{P}(\Gamma_{in})$ und $\mathcal{P}(\Gamma_{out})$ notieren. Da die Eingaben die Umwelteinflüsse auf das System (Lineareinheit) und Ausgaben die Aktionen als Antwort auf die Einflüsse bedeuten, wird das chaotische Modell Ω_{chao} als alle mögliche Systemeinflüsse sowie deren entsprechenden Aktionen dargestellt. Mengentheoretisch lässt sich dies als kartesisches Produkt (Kreuzprodukt) von $\mathcal{P}(\Gamma_{in})$ und $\mathcal{P}(\Gamma_{out})$ darstellen:

$$\Omega_{chao} := \mathcal{P}(\Gamma_{in}) \ \times \ \mathcal{P}(\Gamma_{out})$$

Da alle Systemein- und ausgaben sich um boolesche Signale handeln, lässt sich jedes Szenario als eine boolesche Belegung darstellen. Jeder Wahrheitswert dieser Belegung wird einem Signal zugewiesen, das den Wahrheitswert $True$ oder $False$ annimmt.

Im Weiteren wird die Notation für ein Szenario verwendet, bei der nur das Signal mit dem Wert „$True$“ gezeigt wird. Beispielsweise wird das Szenario

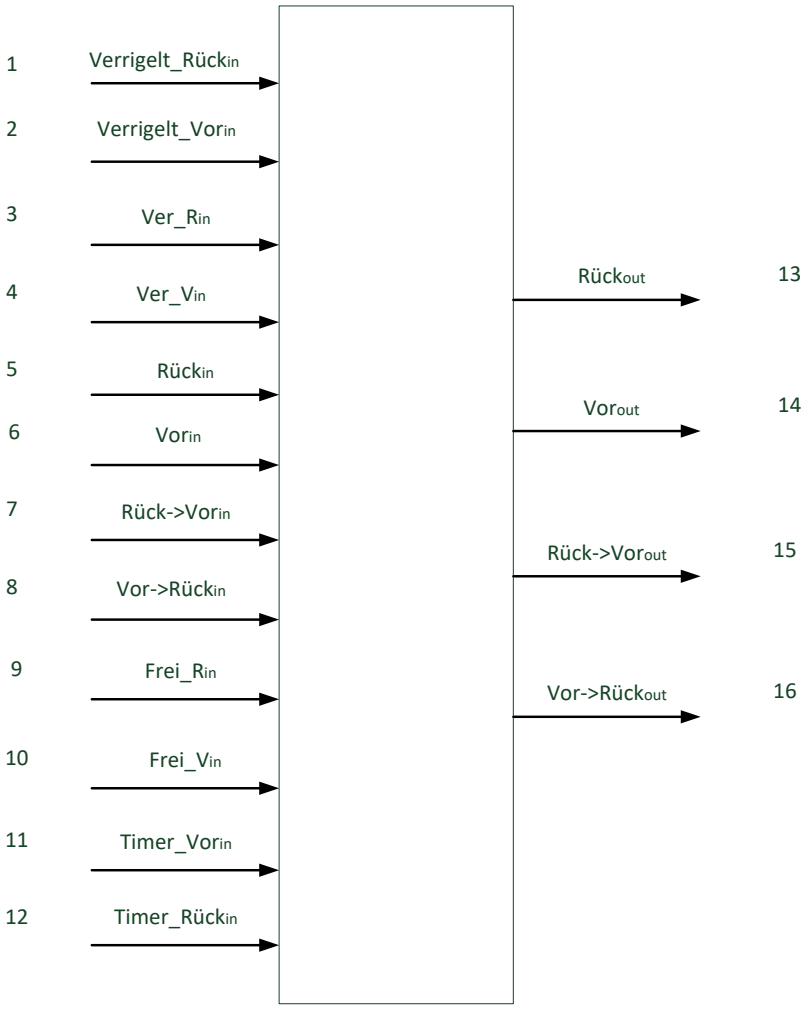


Abbildung 8.8: Systemgrenzenmodell der VEP-Lineareinheit

„Wenn der Zylinder sich auf der Endlage „Vor“ befindet und nicht verriegelt ist und das Freigabe-Signal vorliegt, verlässt der Zylinder die Endlage“

wie folgt dargestellt:

$$\{Ver\_V_{in}, Vor \rightarrow Rück_{in}, Frei\_V_{in}\}, \{Vor \rightarrow Rück_{out}\}$$

Ein fehlerhaftes Szenario kann z.B. sein:

$$\{Ver\_V_{in}, Vor \rightarrow Rück_{in}, Frei\_V_{in}\}, \{Vor_{out}\}$$

welches bedeutet, dass der Zylinder, der sich entriegelt bei der Endlage „Vor“ befindet, kann nicht auf das Freigabe-Signal reagieren und die Endlage verlassen. Ziel der fehlersensitiven Spezifikation liegt darin solches Fehlszenario vollständig zu identifizieren.

Nach der obigen Definition enthält das chaotische Modell alle möglichen Systemverhalten - sowohl die gewünschten fehlerfreien Verhalten, als auch solche Verhaltensanteile, die entweder fehlerbehaftet oder physikalisch nicht interpretierbar sind. Letztere sollen demnächst durch Einführung der Spezifikationsregeln schrittweise aus dem chaotischen Modell entfernt werden.

Das Anwendungsbeispiel „VEP-Lineareinheit“ hat eine Systemgrenze mit 16 Ein- und Ausgangssignalen (12 Eingänge und 4 Ausgänge). Somit enthält das chaotische Modell insgesamt 2^{16} unterschiedliche Szenarien. Mathematica von der Fa. Wolfram Research®, als ein mächtiges CAS (Computer-Algebra-System) - Werkzeug, wird hier eingeführt, um die Datenverarbeitung in solcher großen Menge zu bewältigen (siehe auch [AB14]).

Formulierung der Spezifikationsregeln

Ziel der Fehlerspezifikation liegt darin, aus dem chaotischen Modell die Szenarien zu entfernen, die den Anforderungen widersprechen. Das chaotische Modell wird mit 32 Spezifikationsregeln eingeschränkt. Mit jeder eingeführten Spezifikationsregel wird ein Teil der insgesamt 2^{16} Szenarien des chaotischen Modells eliminiert.

Eine Spezifikationsregel kann z.B. wie folgendes formuliert werden:

$$Vor_{in} \wedge \neg Frei\_R_{in} \Longrightarrow Vor_{out}$$

Sie beschreibt die Systemanforderung, dass wenn der Zylinder sich in der Endlage „Vor“ befindet und das Signal „Frei_R$_{in}$“= $false$ anliegt, bleibt der Zylinder im nächsten

Zeitschritt in der Endlage „Vor“. Unerwünschte Verhalten bezüglich dieser Spezifikationsregel ergeben sich als Komplementrelation bezüglich des chaotischen Modells. Wenn die Elemente dieser Komplementrelation aus dem chaotischen Modell eliminiert werden, bleiben im chaotischen Modell all die Elemente, die die obengenannte Anforderung erfüllen.

Somit entstehen Fehlermodi durch die logische Negation der Spezifikationsregeln. Fehlverhalten, die diese Anforderung verstoßen kann, z.B., dass der Zylinder schon mal die Endlage „Vor“ verlässt, ohne dass das Freigabesignal „Frei_R$_{in}$“ den Wert $True$ aufweist. Dieses Szenario beschreibt ein fehlerhaftes Verhalten, dass der Zylinder ohne Freigabebefehl sich selbst bewegt.

Ein komplementäres Vorgehen dazu wäre, bekannte Fehlermodi exakt zu formulieren, wenn einige davon bekannt sind. Spezifikationsregeln lassen sich dann automatisch durch Negativabbildung der formulieren Fehlermodi ableiten.

Ein naheliegender Fehlermodus ist z. B. wenn die beiden Signale „Frei_V“ und „Frei_R“ gleichzeitig aktiviert sind, was zu einem undefinierten Zustand des Zylinders führen kann.

Anforderung gegen diesen Fehlermodus ist, dass zu einem gewissen Zeitpunkt nur eines von den beiden Signalen den Wert $True$ aufweisen darf. Mit temporaler Logik lässt sich diese Spezifikation wie folgendes formulieren (ff bedeutet hier den logischen Wahrheitswerte $false$):

$$Frei\_V_{in} \wedge Frei\_R_{in} \Longrightarrow ff$$

Prüfung der Verhaltensäquivalenz

Mit jeder hinzugefügten Regel wird eine Menge der Szenarien aus dem chaotischen Modell entfernt. Durch sukzessive Hinzufügung unterschiedlicher Spezifikationsregeln und Eliminierung der Fehlermodi aus dem chaotischen Modell wird eine Situation erreicht, dass es keine weitere Spezifikationsregel gefunden werden kann bzw. kein weiteres fehlerhaftes Szenario aus dem chaotischen Modell entfernt werden kann. D.h. das verbleibend fehlersensitive Modell weist dasselbe Ein- und Ausgangsverhalten wie das ursprüngliche System (intendiertes Verhalten) auf. Das dadurch eliminierte chaotische Modell wird als fehlersensitives Modell gezeichnet. Daher wird diese Situation in [Ort05] als „Verhaltensäquivanlent“ genannt.

Bei diesem Fallbeispiel werden insgesamt 32 Spezifikationsregeln identifiziert, die auch zu 32 Fehlerszenarien korrespondieren. Durch die 32 Eliminationen bleiben im fehlersensitiven Modell noch 20 Szenarien, die alle Spezifikationsregeln erfüllen (siehe Anhang C).

In Tabelle 8.5 werden die 32 Spezifikationsregeln sowie derer entsprechenden Fehlerarten aufgelistet.

Nr.	Spezifikationsregel	Art der Fehlermodi
1	Rück→Vor$_{in}$ ∧ Timer_Vor$_{in}$ ⟹ Vor$_{out}$	Verklemmung
2	Vor→Rück$_{in}$ ∧ Timer_Rück$_{in}$ ⟹ Rück$_{out}$	
3	Verriegelt_Vor$_{in}$∧ Ver_V$_{in}$ ∧ ¬Frei_R$_{in}$ ⟹ Vor$_{out}$	
4	Verriegelt_Rück$_{in}$ ∧ ¬Ver_R$_{in}$ ∧ ¬Frei_V$_{in}$ ⟹ Rück$_{out}$	
5	Rück$_{in}$ ∧ Frei_V$_{in}$ ⟹ Rück→Vor$_{out}$	
6	Vor$_{in}$ ∧ Frei_R$_{in}$ ⟹ Vor→Rück$_{out}$	
7	Verriegelt_Vor$_{in}$ ∧ Ver_V$_{in}$∧ Frei_R$_{in}$ ⟹ Vor→Rück$_{out}$	
8	Verriegelt_Rück$_{in}$ ∧ Ver_R$_{in}$∧ Frei_V$_{in}$ ⟹ Rück→Vor$_{out}$	
9	Vor$_{in}$ ∧ ¬Frei_R$_{in}$ ⟹ Vor$_{out}$	Fälschliches Öffnen/Schließen
10	Rück$_{in}$ ∧ ¬Frei_V$_{in}$ ⟹ Rück$_{out}$	
11	Vor→Rück$_{in}$ ∧ ¬Frei_R$_{in}$ ⟹ Vor→Rück$_{out}$	
12	Rück→Vor$_{in}$ ∧ ¬Frei_V$_{in}$ ⟹ Rück→Vor$_{out}$	
13	Verriegelt_Vor$_{in}$ ∧ ¬Ver_V$_{in}$ ⟹ Vor$_{out}$	Verriegelungsfehler
14	Verriegelt_Rück$_{in}$ ∧¬Ver_R$_{in}$ ⟹ Rück$_{out}$	
15	Verriegelt_Rück$_{in}$ ∧ Ver_R$_{in}$ ⟹ ff	
16	Verriegelt_Vor$_{in}$ ∧ Ver_V$_{in}$ ⟹ ff	
17	Rück$_{in}$ ∧ ¬ Ver_R$_{in}$ ⟹ ff	
18	Vor$_{in}$ ∧ ¬ Ver_V$_{in}$ ⟹ ff	
19	Rück→Vor$_{in}$ ∧ ¬Timer_Vor$_{in}$ ⟹ Rück→Vor$_{out}$	Zeitüberwachungsfehler
20	Vor→Rück$_{in}$ ∧ ¬Timer_Rück$_{in}$ ⟹ Vor→Rück$_{out}$	
21	Timer_Vor$_{in}$ ∧ ¬Rück→Vor$_{in}$ ⟹ ff	
22	Timer_Rück$_{in}$ ∧ ¬Vor→Rück$_{in}$ ⟹ ff	
23	tt ⟹ XOR (Rück$_{out}$, Vor$_{out}$, Rück→Vor$_{out}$, Vor→Rück$_{out}$)	Multizustand
24	¬XOR (Verriegelt_Rück$_{in}$, Verriegelt_Vor$_{in}$, Rück$_{in}$, Vor$_{in}$, Rück→Vor$_{in}$, Vor→Rück$_{in}$) ⟹ ff	
25	¬Ver_R$_{in}$ ∧ Frei_V$_{in}$ ⟹ ff	Störung Freigabekontrolle
26	¬Ver_V$_{in}$ ∧ Frei_R$_{in}$ ⟹ ff	
27	Frei_V$_{in}$ ∧ Frei_R$_{in}$ ⟹ ff	
28	Ver_V$_{in}$ ∧ Ver_R$_{in}$ ⟹ ff	
29	Vor$_{in}$ ∧ Frei_R$_{in}$ ⟹ ff	
30	Rück$_{in}$ ∧ Frei_V$_{in}$ ⟹ ff	
31	¬Frei_R$_{in}$ ∧ Timer_Rück$_{in}$ ⟹ ff	Fälschliches Öffnen/Schließen
32	¬Frei_V$_{in}$ ∧ Timer_Vor$_{in}$ ⟹ ff	

Tabelle 8.5: Spezifikationsregeln für das Fallbeispiel „VEP-Lineareinheit“

8.2.1.3 Abbildung des Störverhaltens in WinMOD

Die im letzten Abschnitt identifizierten Fehlermodi sollen simulativ abgebildet werden. Es ist zu achten, dass nicht alle Fehlermodi eine physikalische Bedeutung erhalten bzw. für den Steuerungstest sinnvoll sind. Die Auswahl der zu betrachtenden Fehlermodi hängt von den konkreten Anwendungsszenarien ab.

Folgende Fehlermodi werden betrachtet und in WinMOD als Störverhalten eingebracht:

- Verriegelungsfehler für Rück
- Verriegelungsfehler für Vor
- Endlage Rück nicht verlassen
- Endlage Vor nicht verlassen
- Endlage Rück verlassen ohne Ansteuerung
- Endlage Vor verlassen ohne Ansteuerung
- Zeitüberwachung Bewegung Rück
- Zeitüberwachung Bewegung Vor
- Störung Freigabekontrolle Rück
- Störung Freigabekontrolle Vor
- Verklemmung Rück
- Verklemmung Vor

Im Steuerungsfunktionsbaustein „VEP-Lineareinheit“ wird das Störmeldungssignal „Stoexx“ True, wenn mindestens einer der vorherigen Fehler ansteht.

Jeder Fehler wird als ein Skript im WinMOD-Element „Force Machine“ implementiert. Die Force Machine ermöglicht die Manipulation von WinMOD-Elementen über das Skript, das mit einem einfachen Syntax Befehle wie „WAIT“, „CMD“ unterstützt. Nähere Beschreibung zur Syntax von WinMOD-Skript wird auf [Mew14] S. 4-19 verwiesen.

Ein Beispiel für den Fehlerfall „Endlage Rück nicht verlassen“ wird in Abbildung 8.9 demonstriert. Die Zifferauswahl am Eingang des WinMOD-Makros aktiviert das Signal „test1“. Die Voraussetzungen für die Stimulation des Fehlerfalls werden durch die beiden anderen Signale „SE11R“ und „KYP11_14“ überprüft. Wenn alle Bedingungen erfüllt sind, wird das Skript für Stimulation des Fehlerverhaltens ausgeführt.

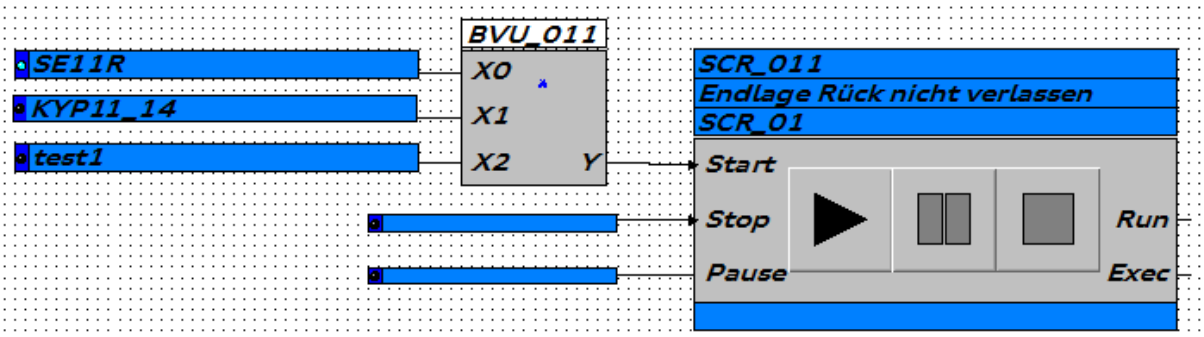


Abbildung 8.9: Umsetzung des Fehlerfalls „Endlage Rück nicht verlassen“in WinMOD

8.2.1.4 Zwischenfazit

Mit dem in diesem Abschnitt vorgestellten Vorgehen ist eine systematische Identifikation der Störverhalten der betrachteten Simulationsmodelle möglich. Im Vergleich zu dem übrigen Vorgehen, bei dem die Störverhalten durch Erfahrungen der Modellersteller auszudenken sind, können die neue Methode die Vollständigkeit sicherstellen. Die Störszenarien können anschließend durch Integration in das Simulationsmode im Zusammenwirken mit der Steuerung testbar.

8.2.2 Testwerkzeug

8.2.2.1 Kommunikation zwischen Testwerkzeug, Simulationswerkzeug und SPS

Wie schon in Kapitel 7 eingeführt, ist es sinnvoll ein Testwerkzeug zu entwickeln, die eine automatische Durchführung der Testfälle in große Anzahl ermöglicht. Außerdem kann das Testwerkzeug Assistenzfunktion zur Testdurchführung anbieten, so dass ein Tester, der nicht tiefe Kenntnisse in Simulation und Steuerungstechnik besitzt, sich auf den Inhalt des Tests konzentrieren kann.

Das in Kapitel 7 entwickelte Konzept und Architektur eines Testframeworks soll in Form eines Testwerkzeugs implementiert werden, welches in der Lage sein soll, gleichzeitig mit der Simulationsumgebung und Steuerung zu kommunizieren. Als Simulationsumgebung wird hier beispielhaft WinMOD von der Firma Mewes & Partner verwendet. Bei der Steuerung handelt es sich um eine SPS mit dem Modell $FC470SPN3TX$ von der Firma PhoenixContact.

Um den Datenaustausch zwischen Testwerkzeug und virtuellem Modell der Anlage bzw. Komponente, also der Simulationsumgebung WinMOD, zu ermöglichen, bietet WinMOD den Schnittstellentreiber Y200 (shared memory) an. Für die Kommunikation zwischen dem Testwerkzeug und speicherprogrammierbaren Steuerungen wird die Schnittstelle OPC

DA betrachtet (vgl. Abbildung 8.10). Im Folgenden werden die beiden Schnittstellen kurz vorgestellt.

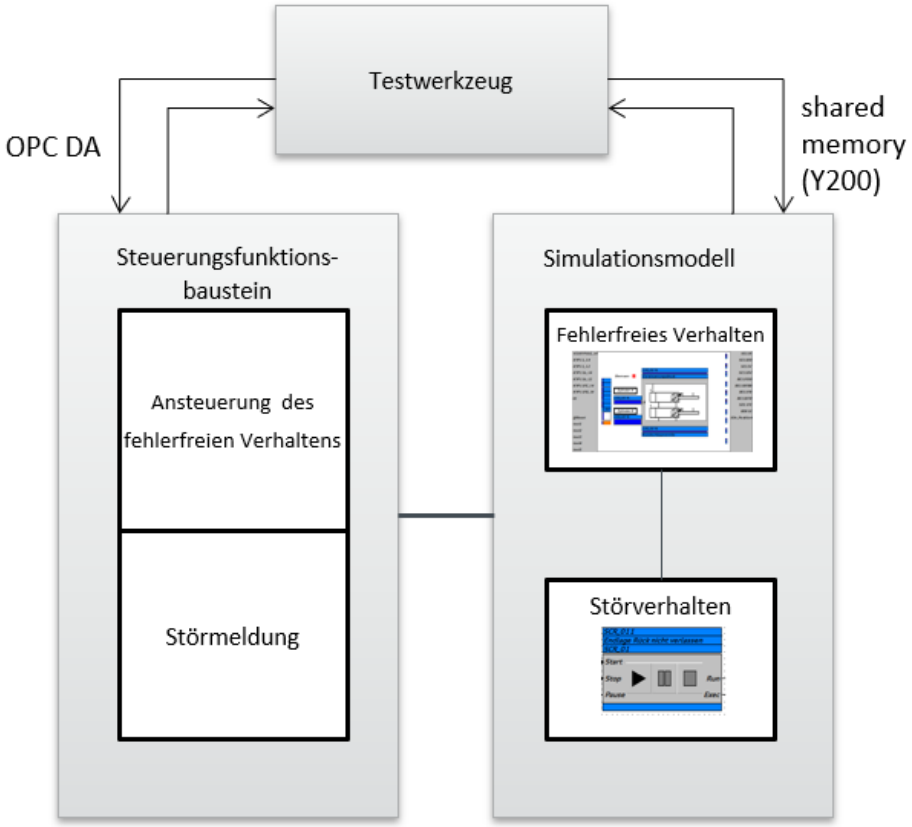


Abbildung 8.10: Strukturelle Darstellung eines VIBN-Arbeitsplatzes mit Testwerkzeug

8.2.2.2 Y200-Schnittstelle (shared memory)

Die Schnittstelle Y200 ermöglicht eigene Anwendungen mit dem WinMOD-System zu verbinden. Dadurch ist es möglich, dass Daten zwischen externen Anwendungen und einem WinMOD-Projekt über einen gemeinsam genutzten Speicherbereich (shared memory) ausgetauscht werden können. Ebenso ist eine physische Trennung zwischen der Fremdapplikation und dem WinMOD-System möglich. Neben dem Schnittstellentreiber ist es möglich, mit dem mitgelieferten Software Development Kit (SDK) eigene Anwendungen zu entwickeln. Diese Anwendungen sind dann in der Lage, über den Schnittstellentreiber Y200 mit WinMOD zu kommunizieren.

8.2.2.3 OPC DA

OPC steht für OLE for Process Control, wobei OLE als Abkürzung für Object Linking and Embedding verwendet wird [Kom09]. Mit OPC ist eine standardisierte, herstellerunabhängige Schnittstelle zwischen OPC-fähigen Applikationen (bspw. Excel, Visual Basic) und

Automatisierungsgeräten (bspw. SPSen, Regler, etc.) gemeint, welche von der OPC Foundation festgelegt wird. Bei dieser Schnittstelle handelt es sich um eine Software-Schnittstelle, welche auf dem von Microsoft entwickelten COM/DCOM basiert und den Zugang zu den Prozessdaten von Automatisierungsgeräten ermöglicht. Die Software-Schnittstelle besteht aus zwei Komponenten, dem OPC-Server und den OPC-Clienten. Die Kommunikation zwischen beiden Komponenten basiert auf dem Client-Server-Prinzip. Wegen der verfügbaren Steuerungsversion wird hier OPC Data Access (OPC DA), die die erste Spezifikation von OPC ist, für die weitere Implementierung ausgewählt. OPC DA ermöglicht einen Zugriff auf die Prozessdaten von SPSen. Mit dieser Spezifikation wird eine Schnittstelle zur Prozessdatenkommunikation zwischen Server- und Clientanwendungen definiert. Somit ist es möglich Daten aus einer Steuerung, mittels dieser Schnittstellendefinition, in einer eigenen Anwendung zu verwenden.

8.2.2.4 Prototypische Entwicklung eines Testwerkzeugs

Nachdem die real möglichen Fehler als ein Skript im WinMOD-Projekt implementiert sind, soll mit dem Testwerkzeug das Verhalten des Steuerungsfunktionsbausteins im Fehlerfall überprüft werden. Das Testwerkzeug wurde mit „Microsoft Visual Studio Ultimate 2010“ mit C++ programmiert und bietet folgende Funktionen an.

- Testwerkzeugschnittstelle zum Simulationswerkzeug (zum WinMOD-Projekt über Y200)
- Testwerkzeugschnittstelle zur SPS (über OPC DA)
- Auswahl der Testfälle aus Testfallbibliothek (Testfälle hier als Signalfolgen hinterlegt in einer Tabelle)
- Präambel- und Postambelfunktionen für die Testfälle, um Komponente in einen definierten Zustand zu bringen
- Aktivierung von Fehlerfällen im Simulationsmodell
- Fehlerauswertung (Vergleich der Ist-Fehlermeldung des Steuerungsprogramms mit der Soll-Fehlermeldung)
- Quittierungsfunktion der Störung im Verhaltensmodell mit Überprüfung der Fehlerbehebung im Steuerungsprogramm

In Abbildung 8.11 ist ein Prototyp des Testwerkzeugs zu sehen, welcher die genannten Funktionen dem Anwender zur Verfügung stellt.

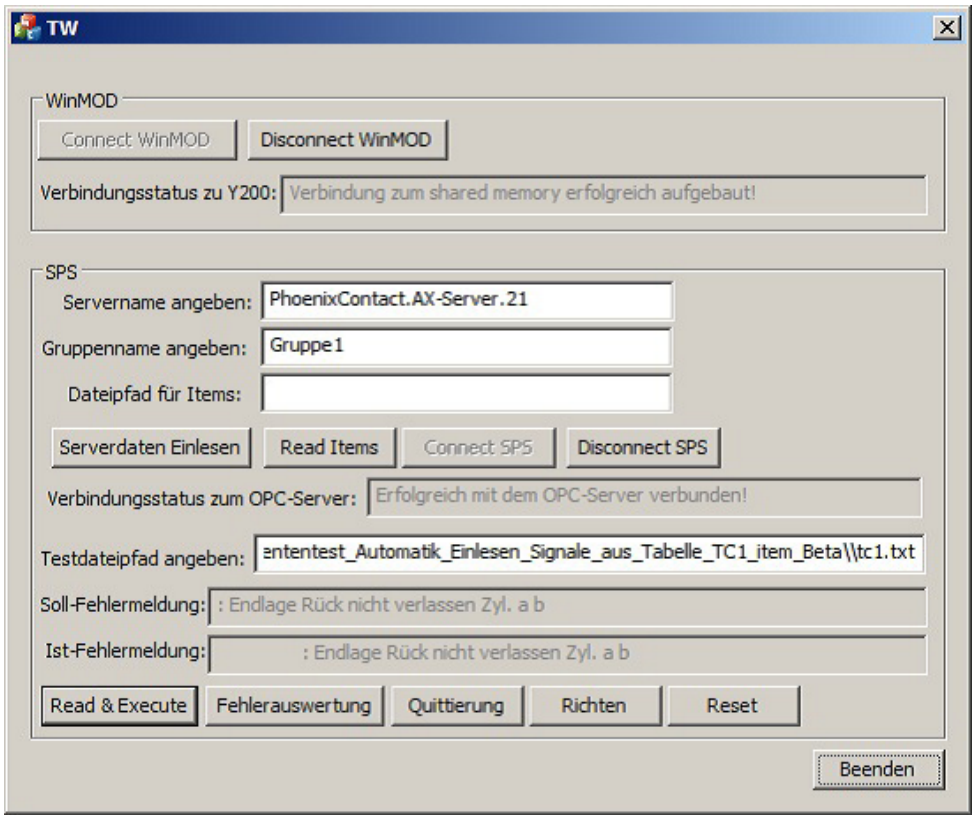


Abbildung 8.11: Testwerkzeug nach Durchführung des Testfalles „Endlage Rück nicht verlassen“

8.2.2.5 Durchführung eines beispielhaften Tests

Der Ablauf der Testsequenz ist in Abbildung 8.12 ersichtlich. Zu Beginn des Tests werden alle Signale, welche im Test genutzt werden, zurückgesetzt. Anschließend erfolgt die Initialisierung des WinMOD-Projekts und des Steuerungsfunktionsbausteins der „VEP-Lineareinheit“. Initialisierung bedeutet, dass im WinMOD-Makro der Testcase ausgewählt und im Steuerungsfunktionsbaustein die Automatik freigegeben und Störungen aktiviert werden. Danach wird die eigentliche Testaktivität durchgeführt. Das heißt, die im Testfall angegebenen Signalfolgen werden abgearbeitet. Dies wird durch die Schaltfläche „Aktion stimulieren Read & Execute“ aktiviert. Bei der Stimulation der Aktion steuert der Steuerungsfunktionsbaustein das WinMOD-Projekt der Komponente an, welche durch die Auswahl der Testfälle absichtlich für die Steuerung fehlerhafte Signale liefert. Dadurch wird erreicht, dass der Steuerungsfunktionsbaustein der Komponente eine Fehlermeldung generiert, welche im Testwerkzeug für die Fehlerauswertung eingelesen wird. Mit der Fehlerauswertung wird das WinMOD-Projekt der Komponente zurückgesetzt. Mittels „Quittierung“ und „Richten“ wird die Fehlermeldung des Steuerungsfunktionsbausteins zurückgesetzt und die Komponente in einen definierten Zustand gebracht.

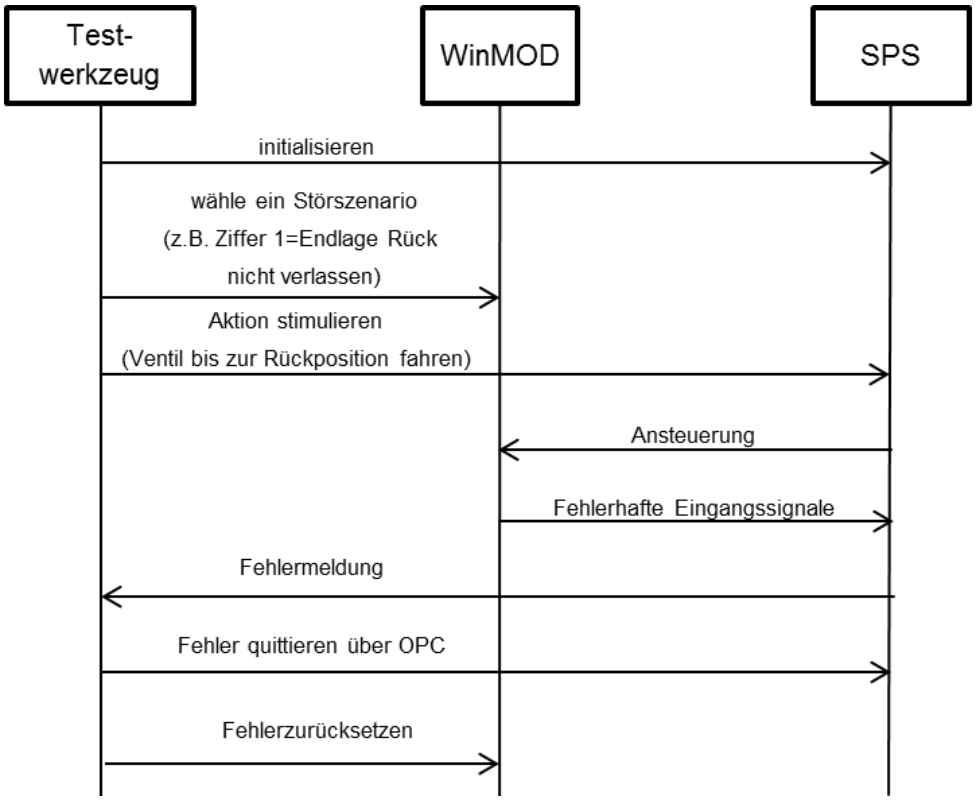


Abbildung 8.12: Sequenzdiagramm des Testfalls: „Endlage Rück nicht verlassen“

8.2.3 Zwischenfazit

In diesem Kapitel wird zunächst aufgezeigt, dass die Abbildung des fehlerhaften Verhaltens einer Komponente bzw. Anlage einen essenziellen ergänzenden Teil darstellt, um das Verhalten des Steuerungsprogramms und die gegebenenfalls umgesetzten Störmeldungen im SPS-Programm zu testen. In einem Anwendungsbeispiel für eine Lineareinheit-Komponente wird das ursprüngliche Verhaltensmodell um das Störverhalten ergänzt. Somit ist das erweiterte Modell in der Lage, den Steuerungsfunktionsbaustein derselben Komponente vollständig zu testen. Jedes Störszenario wird als ein Skript im WinMOD-Element „Force Machine“ implementiert, was eine Trennung zum ursprünglichen fehlerfreien Modell ermöglicht. Darüber hinaus wird der Modellierungsaufwand des Störverhaltens klein gehalten wegen der einfachen Syntax der Skriptsprache. Die Ableitung der Störszenarien erfolgt momentan manuell, was gewisse Erfahrungen und Verständnis mit den Komponenten und Anlagen voraussetzt. Im nächsten Schritt soll untersucht werden, wie die Fehlerszenarien modellbasiert und vollständig abgeleitet werden können.

Um den Testprozess außerdem zu automatisieren, wird ein prototypisches Testwerkzeug entwickelt. Das Testwerkzeug ist in der Lage, sowohl fehlerfreie als auch Störverhalten zu testen. Durch ein Template sind standardisierte Eingabe sowie systematische Verwaltung aller Testdaten möglich. Weitere Assistenzfunktionen bietet das Testwerkzeug

ebenfalls wie die Präambel- und Postambelfunktionen für die Testfälle, um Komponente in einen definierten Zustand zu bringen. Dazu gehören beispielsweise Initialisierungen der Ausgangssituationen eines Testfalls und Rücksetzen des getesteten Simulationsmodells nach der Durchführung eines Testfalls. Dies stellt eine sehr wertvolle Maßnahme dar, um den hohen manuellen Testaufwand signifikant zu verringern. Die Weiterentwicklung des Testwerkzeugs beinhaltet die weiterführende automatische Durchführung der Testfälle sowie die Integration der anlagenspezifischen Testszenarien.

9 Fazit

In diesem Kapitel werden sämtliche Aspekte der Arbeit abschließend zusammengefasst. Anschließend wird ein Ausblick für weitere Forschungsmöglichkeiten aufgezeigt.

9.1 Zusammenfassung

Die virtuelle Inbetriebnahme spielt heutzutage eine immer stärkere Rolle zur Absicherung der Planung und dem frühzeitigen Test der Steuerungstechnik, Validierung der Zusammenarbeit der unterschiedlichen Komponenten (Betriebsmittel) in einer Anlage. Hauptziel ist die Qualitätsverbesserung und Absicherung der Planungsergebnisse und die Beschleunigung der Inbetriebnahme. Um die Akzeptanz der virtuellen Inbetriebnahme weiter zu steigern, ist es nötig, den Aufwand der Modellentwicklung der Anlage zu senken, sowie die Wiederverwendbarkeit der vorhandenen Modellbibliothek zu erhöhen. Die Erstellung der Komponentenmodelle ist keine triviale Aufgabe, die heutzutage meistens der OEM alleine tragen muss. Das OEM-Know-how über eine Komponente ist oft nur über Handbücher oder Datenblätter möglich, die mit der realen Komponente mitgeliefert werden. Auf diese Weise ist das Wissen über die interne Funktionsweise einer Komponente nur sehr schwer oder gar unmöglich vollständig nachvollziehbar, was für den OEM zeit- und arbeitsaufwendig ist.

Die vorliegende Arbeit verfolgt die Zielsetzung, eine Methodik zur Entwicklung und dem Test der Komponentenmodelle für die virtuelle Inbetriebnahme zu erarbeiten. Angestrebt wird ein neues Vorgehenssmodell für den Entwicklungsprozess der Komponentenmodelle. Die Modelle sollen von Herstellern geliefert und knowhow-geschützt an alle Anwender verteilt werden. Um die Nutzung der Komponentenmodelle zu erhöhen, sollen die Modelle auch in der Lage sein, fehlerhafte Verhalten mit zu testen. Damit der Aufwand bei der virtuellen Inbetriebnahme möglichst gering zu halten, soll die Modellierungstiefe nach dem Anwendungszweck skaliert werden. Dies erspart auch die Rechenleistung bei der Ausführung von virtueller Inbetriebnahme. Außerdem ist zur Testausführung ein entsprechendes Werkzeug notwendig.

Nach der Einführung der Motivation, Ziel und Aufbau dieser Arbeit werden anschließend grundlegende Definitionen und Begriffe wie „System“, „Modelle“, „Modellierung“ und „Sim-

ulation“ aufgeklärt. Da Modellierung zu einer der wichtigsten Technologien der virtuellen Inbetriebnahme gehört, wird die Anlagenmodellierung aus unterschiedlichen Aspekten betrachtet. Die Mechatronik bildet den Grundstock der Modelle, die Einbeziehung von Merkmalen und Merkmalträgern erweist sich als zielführend für Abbildung einer Anlagen mit verschiedenen Sichtweisen. Nach einer kurzen Einführung in Digitale Fabrik gibt es eine Erläuterung des Grundprinzips von virtueller Inbetriebnahme als Schwerpunkt. Derer Beziehung zur Digitalen Fabrik kann ebenfalls diskutiert werden. Es wird anschließend auf die für die virtuelle Inbetriebnahme relevanten Konstellationen wie Mil, SiL und Hil eingegangen.

Im Anschluss wird der aktuelle Forschungsstand analysiert. Für das Verständnis werden die Ansätze der Anlagensimulationsmodellierung sowie des Modellentwicklungsprozesses zunächst aufgezeigt. Typische Methoden zur Reduzierung des Aufwands in den Modellierungsprozess werden ebenso vorgestellt. Da der Modelldetaillierungsgrad nicht nur den Modellierungsaufwand beeinflusst, sondern auch bestimmt, ob die Simulation in Echtzeit wegen der eingeschränkten Rechenleistung lauffähig ist, muss die Problematik Modellierungstiefe aus verschiedenen Literaturquellen näher betrachtet werden. Dazu werden Ansätze der Simulationsmodellvalidierung aufgeführt.

Davon ausgehend werden Defizite und Handlungsbedarf für die Komponentenmodellierung herausgearbeitet und Lösungskonzept als Ziele dieser Arbeit abgeleitet. Dies bildet auch die Gliederung des restlichen Teils dieser Arbeit ab. Ein Vorgehensmodell basiert auf dem modifizierten V-Modell schlägt vor, was eine strukturelle Trennung in Komponenten- und Anlagenebene erfordert. Darauf aufbauend wird ein neuer Geschäftsprozess entwickelt, indem die Verhaltensmodelle der einzelnen Komponenten getrennt von der Anlagenmodellierung bereitgestellt werden sollen. Es wird vorgeschlagen, dass die Komponentenmodelle in der Zukunft von den Komponentenzulieferern erstellt werden sollen.

Für die Verhaltensabbildung einer Komponente werden drei identifizierte Forschungsbedarfe behandelt: 1) Die verfügbaren Informationen für die Modellierung sowie die Anforderungen zu den Modellen sind für diverse Anwendungsszenarien unterschiedlich. Aus diesem Grund werden drei Modellierungsansätze in dieser Arbeit eingeführt, die skalierbare Möglichkeiten für die jeweiligen Szenarien angesichts der Detaillierungsstufe, Modellkomplexität usw. bieten können. Dies wird durch ein Modellierungsbeispiel erläutert, um die verschiedenen Merkmale zu zeigen und zu diskutieren, welcher Modellierungsansatz für die virtuelle Inbetriebnahme und den Test des Steuerungsprogramms am besten geeignet ist. 2) Neben der Modellierung der herkömmlichen Komponenten wird die Modellabbildung für sogenannte „intelligente Komponenten“ erörtert, in denen Basisautomatisierungsfunktionen weiter in die Feldkomponente verlagert sind. Diese aus SPS verlagerte integrierte Steuerungsfunktion soll neben dem Verhalten der Komponente ebenso modelliert werden. 3) Nicht nur das normale Verhalten sondern auch das Fehlerverhalten und der fehlerhafte Betrieb wird bei

der Modellierung der Komponente berücksichtigt. Basierend auf diesen Komponentenmodellen identifiziert der Tester während der virtuellen Inbetriebnahme, ob das System / die Komponente in Fehlerszenarien richtig und sicher reagieren kann. Die Ableitung mit dem Vorgehen „fehlersensitive Spezifikation“ wird beleuchtet, die die Identifizierung der fehlerhaften Verhalten unterstützen kann. Diese Forschungsbedarfe werden in der Arbeit systematisch analysiert und es werden Lösungsvorschlägen erarbeitet.

Die Endanwender ist mit der Tatsache konfrontiert, dass die Komponentenmodelle mit unterschiedlichen Werkzeugen und unterschiedlichen Austauschformaten von den Herstellern bereitgestellt werden. Dies erschwert die Nutzung wesentlich und verursacht zeit- und fehleraufwändige zusätzlich Konvertierungsprozesse. Als eine Lösungsmöglichkeit wird in dieser Arbeit ein neutrales Austauschformat basiert auf erweitertem PLCopen XML vorgeschlagen, nachdem einige Kandidaten zur Lösung der benannten Problematik eingeführt und verglichen werden. Dieser Ansatz ermöglicht den reibungslosen Austausch der Komponentenmodelle zwischen Simulationswerkzeugen durch Import- und Exportfunktion.

Die oben diskutierten Handlungsfeldern dienen zur Optimierung der Vorgehensweise der Testdurchführung. Ein Testframework, dass diese Testkonzepte integriert, wird vorgestellt. Dazu wird der Verlauf der Testdurchführung mit dem Testframework erläutert. Das Testframework erleichtert auf der einen Seite den Vorgang der Testdurchführung. Auf der anderen Seite lassen sich Fehlerszenarien, die in dieser Arbeit identifiziert werden können, durch integrierten Fehlerinjektor des Frameworks simulativ erzeugen und bewerten.

Die Umsetzung des Konzepts erfolgte durch zwei Fallstudien. Die erste Fallstudie zeigte durch Modellierung eines SEW-Frequenzumrichters, wie eine solche Komponente mit integrierter Steuerung modelliert werden können. Zudem wird das Modell mit dem vorgeschlagenen neutralen Format dargestellt. Die Äquivalenz zu dem Modell mit ursprünglichem Format kann nachgewiesen werden. In der zweiten Fallstudie werden zusätzlichen zu den normalen Verhalten einer VEP-Lineareinheit auch die fehlerbehafteten Verhalten abgebildet. Die Ableitung erfolgt durch die Methode, die im vorherigen Kapitel eingeführt wird.

9.2 Forschungsbeiträge

Die Forschungsbeiträge dieser Arbeit werden im Folgenden unter Beachtung der in Abschnitt 3.6 genannten Defizite, die in dieser Arbeit behandelt werden, zusammengefasst.

Lösungsansatz zu Defizit 1 :
Kein geeignetes Vorgehensmodell für den Entwicklungsprozess in der virtuellen

Inbetriebnahme ist vorhanden.

Kapitel 4 führt ein Vorgehensmodell ein, das eine kaskadierte Struktur besitzt und den Entwicklungsprozess der virtuellen Inbetriebnahme in komponenten- und Anlagenebene trennt. Kernergebnis ist der neue Geschäftsprozess, der auf diesem neuen Vorgehensmodell beruht. Die Trennung zwischen Komponenten- und Anlagenentwicklung spiegelt den Prozess für den realen Anlagenbau wider und hat den Vorteil, dass die Komponentenhersteller die Komponentenmodelle mit ihrem Know-how entwickeln und testen können. Während der Anlagenhersteller direkt mit der bereitgestellten Komponentenbibliothek arbeiten und sich mit dem Zusammenbau und Konfiguration der Anlagen konzentrieren können.

Lösungsansatz zu Defizit 2 :
Fehlerverhalten sind nicht mit modelliert.

Ein wesentlicher Beitrag dieser Arbeit ist, anders als bei zurzeit etablierten Simulationsmodellen, in denen nur das Normalverhalten berücksichtigt wird, Fehlerverhalten einer Komponente ebenso abzubilden. Darüber hinaus bietet das Vorgehen „fehlersensitive Spezifikation“ einen systematischen Weg, fehlerhafte Szenarien zu identifizieren. Dies erhöht die Testabdeckung der Steuerungsprogramme deutlich, da ein wesentlicher Bestandteil von Steuerungsprogrammen die Aufgaben haben, ausgehend von den unerwünschten Fehlerfällen, Fehlermeldung zu generieren und potentielle Maßnahmen auf Fehlerfällen einzuleiten. Dieser Teil kann erstens getestet werden, wenn das Fehlerverhalten dementsprechend in Simulationsmodell berücksichtigt wird.

Lösungsansatz zu Defizit 3 :
Es fehlt für verschiedene Detaillierungsgrade ein skalierbarer Modellierungsansatz für unterschiedliche Anwendungsfälle.

Drei Modellierungsarten (Black-Box-, Grey-Box- und White-Box-Modell) werden in dieser Arbeit diskutiert, die steigende Detaillierungsstufen sowie Modellkomplexitäten aufweisen. Vorteil von diesem Ansatz ist, Verhaltensmodell für unterschiedliche Anwendungsfälle skalierbar zu erstellen. Anwendungsbereiche der drei Modelle werden diskutiert und mit Kriterien bewertet.

Lösungsansatz zu Defizit 4 :
Testdurchführung in der virtuellen Inbetriebnahme erfolgt meistens noch manuell.

Um die Durchführung der Tests in der virtuellen Inbetriebnahme möglichst zu automatisieren, werden verschiedene Testkonzepte in dieser Arbeit diskutiert und am Ende ein Testframework entwickelt. Testfälle lassen sich in einer Bibliothek abspeichern und bei Bedarf über den Testassistent ausführen. Durch integrierten Fehlerinjektor des Frameworks

lassen sich Fehlerszenarien in dem Modell stimulieren. Ergebnisse der durchgeführten Tests werden eben durch den Testarbiter bewertet.

9.3 Ausblick

Innerhalb dieser Arbeit wird Methodik zur effizienten Entwicklung und dem Test der Komponentenmodelle für die virtuelle Inbetriebnahme erarbeitet. Jedoch besteht für eine Realisierung dieser Methodik in einigen Punkten noch Handlungsbedarf und Anlass zu weiteren Forschungsaktivitäten. Im Folgenden wird auf weitere Arbeiten einzeln eingegangen.

Unterstützung des neutralen Austauschformats durch virtuelle-Inbetriebnahme-Werkzeuge

Damit das neutrale Dateiformat der Komponentenmodelle verbreitet angewendet wird und sich letztendlich durchsetzen kann, müssen die virtuelle-Inbetriebnahme-Werkzeug Schnittstelle für dieses Austauschformat bereitstellen und es verarbeiten können. D.h. die Werkzeuge für virtuelle Inbetriebnahme sollen Komponentenmodelle dargestellt im neutralen Format einlesen, in proprietäres Format konvertieren und dann mit eigenem Lösungsalgorithmus dementsprechend simulieren.

Testmethodik auf Anlagenebene

Nachdem die Komponentenmodelle bei einem OEM oder Anlagenbauer zu Anlagenmodellen zusammengebaut werden, soll Methodik entwickelt werden, die Störszenarien systematisch ableiten und anschließend Testfälle generieren kann. Idealerweise soll diese Funktion in virtuelle-Inbetriebnahme-Werkzeuge eingebunden werden, damit Anlagenentwicklung und -test lückenlos integriert werden, um die Qualität der Modelle noch zu steigern.

Unternehmenübergreifender Entwicklungsprozess

Das in dieser Dissertation eingeführte Vorgehensmodell schlägt vor, Entwicklungsprozess der Verhaltensmodelle auf Komponenten- und Anlagenebene zu teilen, da die Erstellung der Komponentenmodelle durch Hersteller deutlich effizienter ist. Es muss geklärt werden, wie das Know-how der Komponentenhersteller über Ihre Produkte geschützt werden kann, wenn sie den Anwendern die Komponentenmodelle in einem neutralen Format liefern.

A Vergleich der Bibliothek in WinMOD und IEC 61131-3

Kategorie	Simulationselemente	WinMOD	IEC61131-3
Analoge/Digitale	Addition	+	+
	Substraktion	+	+
	Multiplikation	+	+
	Division	+	+
	Skalierung	+	-
	I-Glied	+	-
	D-Glied	+	-
	PTn-Glied	+	-
	Laufzeitglied	+	-
	Vergleicher	+	+
	FormulaX	+	+
Digitale	Zähler	+	+
	Register	+	+
	Ein-/Ausschaltverzögerung	+	-
Binäre	UND	+	+
	ODER	+	+
	Flankenerkennung	+	+
	RS-Flipflop	+	+
	D-Flipflop	+	-

	Binäre Zeitfunktion	+	-
	Taktgenerator	+	-
Konverter	Analog-Digital	+	-
	Digital-Analog	+	-
	Digital-Binär	+	-
	Binär-Digital	+	-
	Analog-Binär	+	-
	Binär-Analog	+	-
Messtechnik	Grenzwertschalter	+	-
	Grenzwertschalter (Digital)	+	-
	Positionsschalter	+	-
	Positionsschalter (digital)	+	-

Tabelle A.1: Vergleich der Bibliothek in WinMOD und IEC 61131-3

B Darstellung des Funktionsbauteins „I-Glied“ mit PLCopen XML

```
<?xml version="1.0" encoding="utf-8"?>
<project xmlns="http://www.plcopen.org/xml/tc6_0200">
  <fileHeader companyName="" productName="CoDeSys" productVersion="
     CoDeSys V3.5 Patch 3" creationDateTime="2018-10-11T16:40:11.5019971"
      />
  <contentHeader name="Unbenannt1.library" modificationDateTime="
     2018-10-11T16:39:30.3997139">
    <coordinateInfo>
      <fbd>
        <scaling x="1" y="1" />
      </fbd>
      <ld>
        <scaling x="1" y="1" />
      </ld>
      <sfc>
        <scaling x="1" y="1" />
      </sfc>
    </coordinateInfo>
    <addData>
      <data name="http://www.3s-software.com/plcopenxml/projectinformation
         " handleUnknown="implementation">
        <ProjectInformation />
      </data>
    </addData>
  </contentHeader>
  <types>
    <dataTypes />
    <pous>
      <pou name="I_Glied" pouType="functionBlock">
```

```
<interface>
  <inputVars>
    <variable name="X">
      <type>
        <REAL />
      </type>
    </variable>
    <variable name="Tn">
      <type>
        <REAL />
      </type>
      <documentation>
        <xhtml xmlns="http://www.w3.org/1999/xhtml">not TIME</xhtml
            >
      </documentation>
    </variable>
    <variable name="Xs">
      <type>
        <REAL />
      </type>
      <initialValue>
        <simpleValue value="0" />
      </initialValue>
    </variable>
    <variable name="S_Eingang">
      <type>
        <BOOL />
      </type>
    </variable>
    <variable name="out_min">
      <type>
        <REAL />
      </type>
    </variable>
    <variable name="out_max">
      <type>
        <REAL />
      </type>
    </variable>
```

```
    <variable name="wrap_around">
      <type>
        <BOOL />
      </type>
    </variable>
  </inputVars>
  <outputVars>
    <variable name="Y">
      <type>
        <REAL />
      </type>
    </variable>
  </outputVars>
  <localVars>
    <variable name="X_last">
      <type>
        <REAL />
      </type>
    </variable>
    <variable name="init">
      <type>
        <BOOL />
      </type>
    </variable>
    <variable name="last">
      <type>
        <DWORD />
      </type>
    </variable>
    <variable name="tx">
      <type>
        <DWORD />
      </type>
    </variable>
    <variable name="K">
      <type>
        <REAL />
      </type>
    </variable>
```

```
            <variable name="zeit_int">
              <type>
                <DWORD />
              </type>
            </variable>
          </localVars>
        </interface>
        <body>
          <ST>
<xhtml xmlns="http://www.w3.org/1999/xhtml">


FUNCTION_BLOCK POU
VAR_INPUT
        X: REAL;
        Tn: REAL;
        Xs: REAL;
        S_Eingang: BOOL;
        out_min: REAL;
        out_max: REAL;
        wrap_around: BOOL;
END_VAR
VAR_OUTPUT
        Y: REAL;
END_VAR
VAR
END_VAR

tx:=TIME_TO_DWORD(TIME());
K:=1/(Tn*1E3);
IF S_Eingang THEN
        Y:=Xs;
ELSIF NOT init THEN
        init:=TRUE;
        X_last:=X;
ELSE
        Y:=(X+X_LAST)*0.5*DWORD_TO_REAL(tx-last)*K+Y;
        X_last:=X;
END_IF;
last:=tx;
```

```
IF Y>=out_max THEN
       Y:=out_max;
ELSIF Y<=out_min THEN
       Y:=out_min;
END_IF;
IF wrap_around THEN
        IF Y>=out_max THEN
               Y:=out_min;
               X:=out_min;
        END_IF;
END_IF;

</xhtml>
         </ST>
       </body>
     </pou>
   </pous>
  </types>
  <instances>
    <configurations/>
  </instances>
</project>
```

C Fehlersensitives Modell einer Zylinder-Ventil-Kombination

{({Vor→Rück$_{in}$},{Vor→Rück$_{out}$}),
({Rück→Vor$_{in}$},{Rück→Vor$_{out}$}),
({Ver_V$_{in}$, Vor→Rück$_{in}$},{Vor→Rück$_{out}$}),
({Ver_V$_{in}$, Vor→Rück$_{in}$, Frei_R$_{in}$},{Vor→Rück$_{out}$}),
({Ver_V$_{in}$, Vor→Rück$_{in}$, Frei_R$_{in}$, Timer_Rück$_{in}$},{Rück$_{out}$}),
({Ver_V$_{in}$, Rück→Vor$_{in}$},{Rück→Vor$_{out}$}),
({Ver_V$_{in}$, Rück→Vor$_{in}$, Frei_R$_{in}$},{Rück→Vor$_{out}$}),
({Ver_V$_{in}$, Vor$_{in}$},{Vor$_{out}$}),
({Ver_R$_{in}$, Vor→Rück$_{in}$},{Vor→Rück$_{out}$}),
({Ver_V$_{in}$, Vor→Rück$_{in}$, Frei_V$_{in}$},{Vor→Rück$_{out}$}),
({Ver_R$_{in}$, Rück→Vor$_{in}$},{Rück→Vor$_{out}$}),
({Ver_R$_{in}$, Rück→Vor$_{in}$, Frei_V$_{in}$},{Rück→Vor$_{out}$}),
({Ver_R$_{in}$, Rück→Vor$_{in}$, Frei_V$_{in}$, Timer_Vor$_{in}$},{Vor$_{out}$}),
({Ver_R$_{in}$, Rück$_{in}$},{Rück$_{out}$}),
({Verriegelt_Vor$_{in}$},{Vor$_{out}$}),
({Verriegelt_Vor$_{in}$, Ver_R$_{in}$},{Vor$_{out}$}),
({Verriegelt_Vor$_{in}$, Ver_R$_{in}$, Frei_V$_{in}$},{Vor$_{out}$}),
({Ver_R$_{in}$},{Rück$_{out}$}),
({Ver_R$_{in}$, Rück→Vor$_{in}$},{Rück$_{out}$}),
({Ver_R$_{in}$, Rück→Vor$_{in}$, Timer_Rück$_{in}$},{Rück$_{out}$})}

Literaturverzeichnis

[3S-10] 3S-Smart Software Solutions (Hrsg.): *Handbuch „Handbuch für SPS Programmierung mit CoDeSys 2.3“*. 3S-Smart Software Solutions, 2010

[AAB+14] Alpar, Paul ; Alt, Rainer ; Bensberg, Frank ; Grob, Heinz L. ; Weimann, Peter ; Winter, Robert: *Anwendungsorientierte Wirtschaftsinformatik: Strategische Planung, Entwicklung und Nutzung von Informations-und Kommunikationssystemen*. Springer DE, 2014

[AB14] Abell, Martha L. ; Braselton, James P.: *The mathematica handbook*. Academic Press, 2014

[AS09] Alt, Oliver ; Schürr, Andy: *Car-Multimedia-Systeme modell-basiert testen mit SysML*. Springer, 2009

[Aut16] AutomationML consortium (Hrsg.): *Whitepaper AutomationML Part 1 - Architecture and general requirements*. AutomationML consortium, April 2016. `https://www.automationml.org.`. – Abgerufen am: 13.04.2015.

[Bar11] Barth, Mike: *Automatisch generierte Simulationsmodelle verfahrenstechnischer Anlagen für den Steuerungstest*, Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg, Diss., 2011

[Bau05] Bauer, Gerhard: *Ölhydraulik: Grundlagen, Bauelemente, Anwendungen*. Bd. 68. Springer-Verlag, 2005

[BDKS09] Bergert, Martin ; Diedrich, Christian ; Kiefer, Jens ; Schmidgall, Guenter: Semantische Beschreibung mechatronischer Betriebsmittel für die Virtuelle Inbetriebnahme von Roboterzellen. In: *Tagungsband Automation*, 2009

[BEP+82] Boehm, Barry W. ; Elwell, James F. ; Pyster, Arthur B. ; Stuckle, E D. ; Williams, Robert D.: The TRW software productivity system. In: *Proceedings of the 6th international conference on Software engineering* IEEE Computer Society Press, 1982, S. 148–156

[Bes05] Bestmann, Boris: Entwicklung eines Simulationsmodells für eine hydraulische Positioniervorrichtung. In: *Studienarbeit, HAW Hamburg* (2005)

[BGW11] Bracht, Uwe ; Geckler, Dieter ; Wenzel, Sigrid: *Digitale Fabrik: Methoden und Praxisbeispiele.* Springer-Verlag, 2011

[BHH+09] Botaschanjan, Jewgenij ; Hensel, Thomas ; Hummel, Benjamin ; Lindworsky, Alexander ; Zäh, Michael F.: Simulationsmodelle für die virtuelle Inbetriebnahme. In: *ATZproduktion* 2 (2009), Nr. 5-6, S. 18–23

[BK08a] Baier, Christel ; Katoen, Joost-Pieter: *Principles of model checking.* MIT press, 2008

[BK08b] Bringmann, Eckard ; Krämer, Andreas: Model-based testing of automotive systems. In: *Software Testing, Verification, and Validation, 2008 1st International Conference on* IEEE, 2008, S. 485–493

[BKHF09] Bergert, Martin ; Kiefer, Jens ; Hoeme, Stephan ; Fedrowitz, Christian: Einsatz der Virtuellen Inbetriebnahme im automobilen Karosserierohbau–Ein Erfahrungsbericht. In: *Tagungsband der 9. Magdeburger Maschinenbau-Tage* (2009), S. 388–397

[BOA+12] Blochwitz, Torsten ; Otter, Martin ; Akesson, Johan ; Arnold, Martin ; Clauss, Christoph ; Elmqvist, Hilding ; Friedrich, Markus ; Junghanns, Andreas ; Mauss, Jakob ; Neumerkel, Dietmar u. a.: Functional mockup interface 2.0: The standard for tool independent exchange of simulation models. In: *Proceedings of the 9th International MODELICA Conference; September 3-5; 2012; Munich; Germany* Linköping University Electronic Press, 2012, S. 173–184

[Boe88] Boehm, Barry W.: A spiral model of software development and enhancement. In: *Computer* 21 (1988), Nr. 5, S. 61–72

[Bos13] Bossel, Hartmut: *Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme.* Springer-Verlag, 2013

[BPSH+15] Barth, Mike ; Puntel-Schmidt, Philipp ; Hoernicke, Mario ; Oppelt, Mathias ; Wolf, Gerrit ; Stern, Oliver ; Hundt, Lorenz: Methoden und Modelle der Virtuellen Inbetriebnahme - Eine Übersicht der Richtlinienarbeit des GMA FA 6.11. In: *Tagungsband Automation*, 2015

[Bun63] Bunge, Mario: A general black box theory. In: *Philosophy of Science* (1963), S. 346–358

[BW19] Biesinger, Florian ; Weyrich, Michael: The facets of digital twins in production and the automotive industry. In: *2019 23rd international conference on mechatronics technology (ICMT)* IEEE, 2019, S. 1–6

[BZBP09] Bungartz, Hans-Joachim ; Zimmer, Stefan ; Buchholz, Martin ; Pflüger, Dirk: *Modellbildung und Simulation: eine anwendungsorientierte Einführung*. Springer-Verlag, 2009

[CG13] Cellier, Francois E. ; Greifeneder, Jurgen: *Continuous system modeling*. Springer Science & Business Media, 2013

[Des] Dessault Systemes (Hrsg.): *DELMIA V6*. Dessault Systemes, `https://www.3ds.com/de/produkte-und-services/delmia/produkte/delmia-v6/`. – Abgerufen am: 11.07.2017.

[DIN77] DIN: DIN 32541: Betreiben von Maschinen und vergleichbaren technischen Arbeitsmitteln. (1977)

[DIN94] DIN: DIN 19226 Teil 1 „Leittechnik - Regelungstechnik und Steuerungstechnik, Allgemeine Grundbegriffe“. (1994)

[Dra09] Drath, Rainer: *Datenaustausch in der Anlagenplanung mit AutomationML: Integration von CAEX, PLCopen XML und COLLADA*. Springer-Verlag, 2009

[Dro04] Dronka, Sven: *Die Simulation gekoppelter Mehrkörper-und Hydraulik-Modelle mit Erweiterung für Echtzeitsimulation*. Shaker, 2004

[ECG] ECGMP: EU GMP Leitfaden, Annexx11: Computerised systems. In: *European Commission, Luxembourg*

[FA12] Food, US ; Administration, Drug: *Code of federal regulations title 21: part 11 - electronic records; electronic signatures*. 2012

[FHS+17] Fay, Alexander ; Hilderbrandt, Constantin ; Scholz, Andre ; Diedrich, Christian ; Schröder, Tizian ; Dobovy, Marin ; Eck, Christian ; Wiegand, Ralf: Semantik für Industrie 4.0-Systeme - Die Basis für den Informationsaustausch in Industrie 4.0-Anwendungsszenarien / Institut für Automatisierungstechnik / Helmut-Schmidt-Universität. 2017. – Forschungsbericht

[FMRS11] Freitag, Matthias ; Müller, Christiane ; Rusch, Gebhard ; Spreitzer, Thomas: *Projektkommunikation: Strategien für temporäre soziale Systeme*. Springer-Verlag, 2011

[FNMS12] Faltinski, S ; Niggemann, O ; Moriz, N ; Schetinin, N: AutomationML als Grundlage für einen durchgängigen Modellierungs-, Simulations-und Integrationsprozess in der Anlagenplanung. In: *Automation, Branchentreff der Mess-und Automatisierungstechnik* 2171 (2012), S. 371–374

[FSGT15] Frieben, Tanja ; Schneider, Marcel ; Gausemeier, Jürgen ; Trächtler, Ansgar: Virtuelle Inbetriebnahme mit wählbarer Modellierungstiefe: Verkürzung der Entwicklungszeit bis zum Start-of-Production. In: *ZWF Zeitschrift für wirtschaftlichen Fabrikbetrieb* 110 (2015), Nr. 4, S. 227–232

[FT06] Follert, Guido ; Trautmann, Andreas: Emulation intralogistischer Systeme. In: *Simulation in Produktion und Logistik* (2006), S. 521–530

[GAM08] GAMP: Good Automated Manufacturing Practice (GAMP R) Guide for a Risk-Based Approach to Compliant GxP Computerized Systems, 5th edn. / International Society for Pharmaceutical Engineering (ISPE), Tampa, FL. 2008. – Forschungsbericht

[GH99] Günthner, WA ; Haller, M: Wie man den Einsatz der Materialfluß-Simulation vereinfachen kann. In: *Logistik im Unternehmen* (1999), S. 70–72

[GH13] Gail, Lothar ; Hortig, H-P: *Reinraumtechnik*. Springer-Verlag, 2013

[GHR+03] Grabowski, Jens ; Hogrefe, Dieter ; Réthy, György ; Schieferdecker, Ina ; Wiles, Anthony ; Willcock, Colin: An introduction to the testing and test control notation (TTCN-3). In: *Computer Networks* 42 (2003), Nr. 3, S. 375–403

[Gip13] Gipser, Michael: *Systemdynamik und Simulation*. Springer-Verlag, 2013

[Gis15] Gischel, Bernd: *Handbuch EPLAN electric*. Carl Hanser Verlag GmbH Co KG, 2015

[Güh10] Gühmann, Clemens: *Simulation und Test für die Automobilelektronik: Vom Konzept bis zur Serie; mit 12 Tabellen/[Vierte TagungSSimulation und Test in der Funktions-und Softwareentwicklung für die Automobilelektronikïm Mai/Juni 2010]. TU, Berlin; IAV GmbH, Ingenieurgesellschaft Auto und Verkehr. Clemens Gühmann; Thieß-Magnus Wolter (Hrsg.) und 123 Mitautoren*. expert-Verlag, 2010

[Had15] Hadlich, Thomas: *Verwendung von Merkmalen im Engineering von Systemen*, Magdeburg, Universität, Diss., 2015, Diss., 2015

[Hal13] Halang, Wolfgang A.: *Funktionale Sicherheit: Echtzeit 2013*. Springer-Verlag, 2013

[Hel13] Helbing, Dirk: *Verkehrsdynamik: neue physikalische Modellierungskonzepte*. Springer-Verlag, 2013

[HFMW17] Hoang, Xuan-Luu ; Fay, Alexander ; Marks, Philipp ; Weyrich, Michael: Generation and impact analysis of adaptation options for automated manufacturing machines. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* IEEE, 2017, S. 1–8

[HFMW18] Hoang, Xuan L. ; Fay, Alexander ; Marks, Philipp ; Weyrich, Michael: Industrial Application of a MDM-based Approach for Generation and Impact Analysis of Adaptation Options-a Case Study. In: *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)* Bd. 1 IEEE, 2018, S. 1244–1247

[HI99] Harmonization (ICH), International C.: GMP for active pharmaceulical ingradients. In: *Guideline Q7A* 4 (1999)

[Hil12] Hillenbrand, Martin: *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen*, Diss., 2012

[HKW08] Himpel, F. ; Kaluza, B. ; Wittmann, J.: *Spektrum des Produktions- und Innovationsmanagements: Komplexität und Dynamik im Kontext von Interdependenz und Kooperation*. Gabler Verlag, 2008 (Gabler Edition Wissenschaft). – ISBN 9783835055834

[Hof13] Hoffmann, Dirk W.: *Software-Qualität*. Springer-Verlag, 2013

[HRVG81] Haasl, D.F. ; Roberts, N.H. ; Vesely, W.E. ; Goldberg, F.F.: Fault tree handbook. In: *Nuclear Regulatory Commission, Washington, DC (USA). Office of Nuclear Regulatory Research* (1981)

[HSSF17] Hauf, Dominik ; Süss, Sebatian ; Strahilov, Anton ; Franke, Jörg: Multifunctional use of functional mock-up units for application in production engineering. In: *IEEE 15th International Conference on Industrial Informatics (INDIN)*, 2017

[HTF96] Harashima, F ; Tomizuka, M ; Fukuda, T: Mechatronics - What Is It, Why, and How? - An Editorial. In: *IEEE/ASME Transactions on Mechatronics* (1996)

[IEC09] IEC, DIN: *60050-351: Internationales Elektrotechnisches Wörterbuch-Teil 351: Leittechnik*. 2009

[IEC13] IEC: *61131-3: Programmable controllers - Part 3: Programming languages.* 2013

[IEC15] IEC: *62832-1: Industrial-process measurement, control and automation - Digital Factory framework - Part 1: General principles.* 2015

[IEE90] IEEE: IEEE standard glossary of software engineering terminology. In: *New York, USA* (1990)

[Ise07] Isermann, Rolf: *Mechatronische systeme: grundlagen.* Springer-Verlag, 2007

[ISO11] ISO: *ISO 26262: Road vehicles - Functional safety.* 2011

[ITEa] ITEA (Hrsg.): *Avanti, System methodology for virtual commissioning based on behavior simulation of production systems.* ITEA, `http://www.avanti-project.de/results.html`. – Abgerufen am: 01.02.2019.

[ITEb] ITEA (Hrsg.): *Entoc, Engineering Toolchain.* ITEA, `https://entoc.eu/`. – Abgerufen am: 01.02.2019.

[Jen01] Jenny, Bruno: *Projektmanagement in der Wirtschaftsinformatik.* vdf Hochschulverlag AG, 2001

[JJK+20] Jung, Tobias ; Jazdi, Nasser ; Krauss, Stefan ; Köllner, Christian ; Weyrich, Michael: Hardware-in-the-loop simulation for a dynamic co-simulation of internet-of-things-components. In: *Procedia CIRP* 93 (2020), S. 1334–1339

[JJW17] Jung, Tobias ; Jazdi, Nasser ; Weyrich, Michael: A survey on dynamic simulation of automation systems and components in the Internet of Things. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* IEEE, 2017, S. 1–4

[JSW18] Jung, Tobias ; Shah, Payal ; Weyrich, Michael: Dynamic co-simulation of internet-of-things-components using a multi-agent-system. In: *Procedia CIRP* 72 (2018), S. 874–879

[Kah13] Kahlbrandt, Bernd: *Software Engineering: Objektorientierte Software-Entwicklung mit der Unified Modeling Language.* Springer-Verlag, 2013

[KB16] Kamiske, G.F. ; Brauer, J.P.: *Qualitätsmanagement von A - Z: Wichtige Begriffe des Qualitätsmanagements und ihre Bedeutung.* Carl Hanser Verlag GmbH & Company KG, 2016. – ISBN 9783446447523

[KBR10] Kiefer, Jens ; Bergert, Martin ; Rossdeutscher, Mario: Mechatronic Objects in Production Engineering. In: *atp edition-Automatisierungstechnische Praxis* 52 (2010), Nr. 12, S. 36–44

[Kie07] Kiefer, Jens: *Mechatronikorientierte Planung automatisierter Fertigungszellen im Bereich Karosserierohbau*, Universität des Saarlandes, Diss., 2007

[KOB09] Kiefer, Jens ; Ollinger, Lisa ; Bergert, Martin: Virtuelle Inbetriebnahme-Standardisierte Verhaltensmodellierung mechatronischer Betriebsmittel im automobilen Karosserierohbau. In: *Automatisierungstechnische Praxis* 51 (2009), Nr. 7, S. 40–46

[Kom09] Kominek, Darek: OPC - was ist das eigentlich? „Das OPC-Handbuch für Jedermann“. (2009)

[Kra07] Krause, Herbert: Virtual commissioning of a large LNG plant with the DCS 800XA by ABB. In: *6th EUROSIM Congress on Modelling and Simulation, Ljubljana, Slovénie*, 2007

[Kra11] Krause, J: *Testfallgenerierung aus modellbasierten Systemspezifikationen auf der Basis von Petrinetzentfaltungen. Aachen*, Diss., 2011

[Kuf12] Kufner, A: *Automatisierte Erstellung von Maschinenmodellen für die Hardware-inthe-Loop-Simulation von Montagemaschinen*, Universität Stuttgart, Diss., 2012

[LBD+15] Liu, Zheng ; Bieliaiev, Oleksandr ; Diedrich, Christian ; Meyer, Torben ; Völzke, Benjamin: Komponentenmodelle für die Virtuelle Inbetriebnahme. In: *Tagungsband Automation*, 2015

[LD12] Liu, Zheng ; Diedrich, Christian: Validierungskonzept für virtuelle Anlagen. In: *Tagungsband Automation*, 2012

[LDGSD15] Liu, Zheng ; Dr. Gunar, Hartung ; Seidel, Chris ; Diedrich, Christian: Konzept und Entwicklung eines Testwerkzeuges für die Virtuelle Inbetriebnahme. In: *Tagungsband der 12. Magdeburger Maschinenbau-Tage* (2015)

[Lin08] Lindworsky, Alexander: Virtuelle Inbetriebnahme–Nutzenmaximierung durch Automatisierung der Modellerstellung. In: *Internationales Forum Mechatronik*, 2008, S. 412–425

[Lin11] Lindworsky, Alexander: *Teilautomatische Generierung von Simulationsmodellen für den entwicklungsbegleitenden Steuerungstest*, Fakultät für Maschinenwesen der Technischen Universität München, Diss., 2011

[LSC12] *Kapitel* Virtual Commissioning of Automated Systems. In: Liu, Zheng ; Suchold, Nico ; Christian, Diedrch: *Automation*. Intech Open, 2012, S. 131–148

[Mes17] Mes, Martijn: Simulation Modelling using Practical Examples: A Plant Simulation Tutorial. (2017)

[Mew05] Mewes, Jürgen: Virtuelle Inbetriebnahme mit realen Automatisierungssystemen und virtuellen Maschinen. In: *Hennigsdorf: Mewes* (2005)

[Mew14] Mewes & Partner GmbH (Hrsg.): *WinMOD-Systemsoftware V7.1 Handbuch*. Mewes & Partner GmbH, 2014

[Mey14] Meyer, Torben: *Wirtschaftliche Erstellung von Emulationsmodellen für die virtuelle Inbetriebnahme*, Ilmenau, Technische Universität, Diss., 2014

[MHWF18] Marks, Philipp ; Hoang, Xuan L. ; Weyrich, Michael ; Fay, Alexander: A systematic approach for supporting the adaptation process of discrete manufacturing machines. In: *Research in Engineering Design* 29 (2018), Nr. 4, S. 621–641

[MKD12] Magnus, Stephan ; Krause, Jan ; Diedrich, Christian: Test generation for model based fieldbus profiles. In: *2012 IEEE International Conference on Industrial Technology* IEEE, 2012, S. 682–687

[MKD15] Magnus, Stephan ; Krause, Jan ; Diedrich, Christian: Automatisierte Modellgenerierung auf Basis formalisierter Anforderungsbeschreibung. In: *at-Automatisierungstechnik* 63 (2015), Nr. 2, S. 121–128

[Mod] Modelica Association (Hrsg.): *Functional Mock-up Interface*. Modelica Association, `www.fmi-standard.org`. – Abgerufen am: 15.07.2018.

[MOD14] MODELISAR: *Functional Mock-up Interface for Model Exchange and Co-Simulation*, 2014

[MRKW16] Magnus, Stephan ; Russ, Tim ; Krause, Jan ; Wissenschaftshafen, Denkfabrik im: Anforderungs-und modellbasierte Testfallgenerierung und Testdurchführung unter Nutzung von Methoden zur Netzwerkanalyse. In: *14. Fachtagung EKA - Entwurf komplexer Automatisierungssysteme*, Otto-von-Guericke-Universität Magdeburg/ifak, May 2016

[MW09] Mewes, Jürgen ; Wegener, Friedrich: Virtuelle Inbetriebnahme von Förderanlagen und Materialflusssimulation. In: *Tagungsband Automation*, 2009

[OBU15] Oppelt, Mathias ; Barth, Mike ; Urbas, Leon: The role of simulation within the life-cycle of a process plant. In: *Results of a global online survey* (2015)

[OR04] Ortmeier, Frank ; Reif, Wolfgang: Failure-sensitive specification: A formal method for finding failure modes. (2004)

[Ort05] Ortmeier, Frank: *Formale Sicherheitsanalyse*, Universität Augsburg, Diss., 2005

[OWBU15] Oppelt, Mathias ; Wolf, Gerrit ; Barth, Mike ; Urbas, Leon: Simulation im Lebenszyklus einer Prozessanlage. In: *atp edition* 57 (2015), Nr. 09, S. 46–59

[P+00] Pemberton, Steven u. a.: XHTML 1.0 the extensible hypertext markup language. In: *W3C Recommendations* (2000), S. 1–11

[Pie17] Piepenbrock, Dr. G.: Virtual Commissioning in Plant Simulation utilizing the new SIMATIC S7-PLCSIM Advanced Interface and OPC UA. In: *2017 Plant Simulation Worldwide User Conference*, 2017

[PLC08] PLCopen Technical Committee 6 (Hrsg.): *Technical Paper XML Formats for IEC 61131-3*. PLCopen Technical Committee 6, April 2008

[PS16] Puntel Schmidt, Philipp: *Methoden zur simulationsbasierten Absicherung von Steuerungscode fertigungstechnischer Anlagen*, Universität der Bundeswehr Hamburg, Diss., 2016

[Ras90] Rasmussen, Jens: A model for the design of computer integrated manufacturing systems: identification of information requirements of decision makers. In: *International Journal of Industrial Ergonomics* 5 (1990), Nr. 1, S. 5–16

[RBGW12] Rossner, Thomas ; Brandes, Christian ; Götz, Helmut ; Winter, Mario: *Basiswissen Modellbasierter Test*. Dpunkt. verlag, 2012

[Rop75] Ropohl, Günter: Einleitung in die Systemtechnik. In: *Systemtechnik–Grundlagen und Anwendung, München Wien, Carl Hanser Verlag* (1975), S. 1–77

[Ros11] Rossdeutscher, Mario: *Entwicklung eines Verfahrens zum Programmtest in der robotergestützten Montage*, Lehrstuhl Automatisierungstechnik, BTU Cottbus, Diss., 2011

[Roy70] Royce, Winston W.: Managing the development of large software systems. In: *proceedings of IEEE WESCON* Bd. 26 Los Angeles, 1970, S. 328–388

[SBMS10] Strassburger, Steffen ; Bergmann, Sören ; Müller-Sommer, Hannes: Modellgenerierung im Kontext der Digitalen Fabrik-Stand der Technik und Herausforderungen. In: *Integrationsaspekte der Simulation: Technik, Organisation und Personal. Karlsruhe: KIT Scientific Publishing* (2010), S. 37–44

[Sch07] Schob, U.: *Mechatronische Modellbildung von Produktionsanlagen: Konzepte und Methoden zur virtuellen Inbetriebnahme durch Softwarekopplung von Entwicklungswerkzeugen.* VDM, Müller, 2007. – ISBN 9783836450928

[SD18] Süss, Sebastian ; Diedrich, Christian: Klassifizierung mechatronischer Komponenten für Anwendungen der virtuellen Anlagenabsicherung. In: *15. Fachtagung EKA-Entwurf komplexer Automatisierungssysteme, 02.-03.05.2018* ifak/Otto-von-Guericke-Universität Magdeburg, 2018

[See99] Seel, Uwe: *Robotergestützte Zellenkalibrierung als Basis einer featurebasierten Montageplanung.* Saarbrücken, Lehrstuhl für Fertigungstechnik/-CAM der Universittät des Saarlandes, Diss., 1999

[Sei03] Seidewitz, Ed: What models mean. In: *IEEE software* (2003), Nr. 5, S. 26–32

[Sei13] Seitz, Matthias: Prozesssimulation im Automatisierungssystem. In: *atp edition* 55 (2013), Nr. 11, S. 26–31

[Sel05] Selke, Carsten: *Entwicklung von Methoden zur automatischen Simulationsmodellgenerierung*, Institut für Werkzeugmaschinen und Betriebswissenschaften (iwb), Technische Universität München, Diss., 2005

[SEW10a] SEW-EURODRIVE (Hrsg.): *Handbuch „Systemhandbuch MOVIDRIVE® Buspositionierung“.* SEW-EURODRIVE, 07/2010

[SEW10b] SEW-EURODRIVE (Hrsg.): *Handbuch „Systemhandbuch MOVIDRIVE® MDX60B/61B“.* SEW-EURODRIVE, 09/2010

[Sha75] Shannon, Robert E.: *Systems simulation: the art and science.* Bd. 1. Prentice-Hall Englewood Cliffs, NJ, 1975

[Sie09] Siemens AG (Hrsg.): *SIMIT SCE.* Siemens AG, 2009

[SKB05] Schmidgall, Günter ; Kiefer, Jens ; Bär, Thomas: Objectives of integrated digital production engineering in the automotive industry. In: *Proceedings of the 16th IFAC World Congress, Prague, Czech Republic*, 2005

[SLT13] Schmüdderrich, T ; LOCHBIECHLER, M ; Trächtler, A: Methodik zur anforderungsgerechten Wahl der Modellierungstiefe von Verhaltensmodellen für die virtuelle Inbetriebnahme. In: *Fachtagung Mechatronik* 6 (2013), Nr. 8

[SMT+16] Süss, Sebastian ; Magnus, Stephan ; Thron, Mario ; Zipper, Holger ; Odefey, Ulrich ; Fässler, Victor ; Strahilov, Anton ; Kłodowski, Adam ; Bär, Thomas ; Diedrich, Christian: Test methodology for virtual

commissioning based on behaviour simulation of production systems. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)* IEEE, 2016, S. 1–9

[Spi09] Spitzweg, Michael: *Methode und Konzept für den Einsatz eines physikalischen Modells in der Entwicklung von Produktionsanlagen*, Technische Universität München, Diss., 2009

[Spl96] Splanemann, Ralph: *Teilautomatische Generierung von Simulationsmodellen aus systemneutral definierten Unternehmensdaten am Beispiel der Integration der Materialflusssimulation in die Planung von Fertigungsanlagen*, Universität Bremen, Diss., 1996

[SS13] Seitz, Matthias ; Steck, Thorsten: Flexibles, automatisiertes Simulieren und Testen in speicherprogrammierbaren Steuerungen. In: *A&D-Kompendium 2013/14*, publish-industry Verlag, 2013

[SSD15] Süss, Sebastian ; Strahilov, Anton ; Diedrich, Christian: Behaviour simulation for virtual commissioning using co-simulation. In: *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)* IEEE, 2015, S. 1–8

[Sta73] Stachowiak, Herbert: Allgemeine Modelltheorie. (1973)

[Tom98] Tomaszunas, Jerzy J.: *Komponentenbasierte Maschinenmodellierung zur Echtzeit-Simulation für den Steuerungstest.* Utz, Wiss., 1998

[Tra06] Tracht, Kirsten: Verbundprojekt Ramp-Up/2. In: *Universität Hannover* (2006)

[VDI00] VDI: Richtlinie 3633 Blatt 1: Simulation von Logistik-, Materialfluss- und Produktionssystemen - Grundlagen. In: *Materialfuss- und Produktionssystemen-Grundlagen, Düsseldorf: VDI-Verlag* (2000)

[VDI04] VDI: Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme. In: *Düsseldorf: VDI-Verlag* (2004)

[VDI07] VDI: Richtlinie 3633 Blatt 8: Simulation von Logistik-, Materialfluss- und Produktionssystemen - Teil 8: Maschinennahe Simulation. (April 2007)

[VDI08] VDI: Richtlinie 4499, Blatt 1: Digitale Fabrik - Grundlagen. Entwicklungsmethodik für mechatronische Systeme. In: *Düsseldorf: VDI-Verlag* (2008)

[VDI15a] VDI: Richtlinie 3682 Blatt 1: Formalisierte Prozessbeschreibung - Konzept und grafische Darstellung. (2015)

[VDI15b] VDI: Richtlinie 3693, Blatt 1: Virtuelle Inbetriebnahme - Modellarten und Glossar. In: *Düsseldorf: VDI-Verlag* (2015)

[VDI17] VDI: VDI/VDE-GMA 7.21: Industrie 4.0 - Begriffe/Terms. In: *Düsseldorf. April* (2017)

[VGRH81] Vesely, William E. ; Goldberg, Francine F. ; Roberts, Norman H. ; Haasl, David F.: Fault tree handbook / Nuclear Regulatory Commission Washington dc. 1981. – Forschungsbericht

[VSW+18] Vialkowitsch, Jens ; Schell, Otto ; Willner, Alexander ; Vollmar, Friedrich ; Schulz, Thomas ; Pethig, Florian ; Neidig, Jörg ; Usländer, Thomas ; Reich, Johannes ; Nehls, Daniel ; Lieske, Matthias ; Diedrich, Christian ; Belyaev, Alexander ; Bock, Jürgen ; Deppe, Torben: *I4.0-Sprache Vokabular, Nachrichtenstruktur und semantische Interaktionsprotokolle der I4.0-Sprache.* Bundesministerium für Wirtschaft und Energie (BMWi), April 2018

[Web13] Weber, Klaus H.: *Inbetriebnahme verfahrenstechnischer Anlagen: Praxishandbuch mit Checklisten und Beispielen.* Springer-Verlag, 2013

[Wes03] Westkämper, Engelbert: Die Digitale Fabrik. In: *Neue Organisationsformen im Unternehmen: Ein Handbuch für das moderne Management. Berlin, Heidelberg: Springer* (2003), S. 788–797

[WS00] Whittle, Jon ; Schumann, Johann: Generating statechart designs from scenarios. In: *Proceedings of the 22nd international conference on Software engineering* ACM, 2000, S. 314–323

[WS02] Whittle, Jon ; Schumann, Johan: statechart synthesis from scenarios: An air traffic control case study. In: *Proc.of „Scenarios and State-Machines: models, algorithms and tools“ workshop at the 24th Int.Conf.on Software Engineering (ICSE 2002)*, 2002

[WSG11] Will, D ; Ströhl, H ; Gebhardt, N: Hydraulik: Grundlagen, Komponenten, Schaltungen. 5., neu bearb. und erw. In: *Aufl. Literaturverz. S.[491]-504. Berlin [ua]: Springer* (2011)

[Wün07] Wünsch, Georg: *Methoden für die virtuelle Inbetriebnahme automatisierter Produktionssysteme*, Technischen Universität München, Diss., 2007

[Xu03] Xu, Lingxiang: *Wiederverwendbare Modelle zur Maschinensimulation für den Steuerungstest.* Utz, Wiss., 2003

[Zeu98] Zeugträger, Karsten: *Anlaufmanagement für Großanlagen.* VDI-Verlag, 1998

[Zip21] Zipper, Holger: *Verfahren zur Synchronisation betriebsparalleler Simulationen durch Online-Parameterschätzung*, Otto-von-Guericke Universität, Diss., 2021

[ZR04] Zäh, Michael ; Reinhart, Gunther: *Virtuelle Produktionssystemplanung: virtuelle Inbetriebnahme und digitale Fabrik, Garching, 23. September 2004.* Utz, 2004