TUM

TECHNISCHE UNIVERSITÄT MÜNCHEN
INSTITUT FÜR INFORMATIK

IMoMeSA - Abschlussbericht

Georg Hackenberg, Christoph Richter, Michael Zäh

TUM-I1519

IMoMeSA – Abschlussbericht

Integrierte modellbasierte Entwicklung mechatronischer Systeme im Maschinen- und Anlagenbau

Das IGF-Vorhaben 435 ZN der Forschungsvereinigung VDW – Verein deutscher Werkzeugmaschinen e.V. wurde über die AiF im Rahmen des Programms zur Förderung der Industriellen Gemeinschaftsförderung (IGF) vom Bundesministerium für Wirtschaft und Energie aufgrund eines Beschlusses des Deutschen Bundestages gefördert.

Georg Hackenberg, M.Sc.
Lehrstuhl für Software & Systems Engineering
Fakultät für Informatik
Technische Universität München

Telefon: +49 (89) 289 - 17080
Fax: +49 (89) 289 - 17307
E-Mail: hackenbe@in.tum.de
Web: http://www4.in.tum.de/~hackenbe/

Christoph Richter, Dipl.-Ing.
Institut für Werkzeugmaschinen und Betriebswirtschaft (iwb)
Fakultät für Maschinenwesen
Technische Universität München

Telefon: +49 (89) 289 - 15547
Fax: +49 (89) 289 - 15555
E-Mail: christoph.richter@iwb.tum.de
Web: http://www.iwb.tum.de/

Michael Zäh, Prof. Dr.-Ing.
Institut für Werkzeugmaschinen und Betriebswirtschaft (iwb)
Fakultät für Maschinenwesen
Technische Universität München

Telefon: +49 (89) 289 - 15502
Fax: +49 (89) 289 - 15555
E-Mail: michael.zaeh@iwb.tum.de
Web: http://www.iwb.tum.de/Zaeh_zh-p-955327.html

Inhaltsverzeichnis

1 Ausgangssituation

Das Forschungsprojekt IMoMeSA (Integrierte modellbasierte Entwicklung Mechatronischer Systeme im Maschinen- und Anlagenbau) widmete sich der Aufgabe, eine Entwicklungsmethodik für mechatronische Systeme zu erarbeiten. Diese soll es ermöglichen, Maschinen und Anlagen vom ersten Grundkonzept bis zu einem virtuellen Prototyp modellbasiert zu entwickeln und mittels einer Simulation abzusichern.

Im Rahmen des ersten Kapitels dieses Abschlussberichtes wird zunächst die *Motivation* für die Erarbeitung modelbasierter Entwicklungsprinzipien mit Bezug zu aktuellen Trends und Herausforderungen des Maschinen- und Anlagenbaus herausgestellt. Darauf aufbauend werden die *Vorarbeiten* aus dem AiF-Forschungsprojekt AutoVIBN vorgestellt, die als unmittelbare Basis für IMoMeSA dienten. Anschließend wird die konkrete *Zielstellung* von IMoMeSA unter Berücksichtigung der einzelnen Arbeitspakete dargelegt, bevor der *Aufbau* dieses Abschlussberichts beschrieben wird.

1.1 Motivation

Vor dem Hintergrund aktueller produktionstechnischer Trends, wie einer zunehmenden Globalisierung oder Ressourcenverknappung, sehen sich Unternehmen des Maschinen- und Anlagenbaus derzeit mit einer Vielzahl an Herausforderungen konfrontiert [1]. Neben technologischen Weiterentwicklungen oder einer fortwährenden Weiterbildung von Fach- und Methodenkompetenzen der Mitarbeiter bedarf es insbesondere effizienterer Arbeitsabläufe und einer verstärkten Unterstützung der Entwicklungsprozesse durch leistungsstarke Softwaretools [2]. Eine Studie der Gesellschaft für Mess- und Automatisierungstechnik [3] bestätigt dies mit der Forderung nach der „Realisierung einer durchgängigen Modellierung“, der „Entwicklung von neuen Methoden zum Informations- und Wissensmanagement“ sowie der „Erarbeitung von neuen Methoden zum systematischen Entwurf komplexer Systeme mit gewünschten Eigenschaften“.

Derzeitige Entwicklungsprozesse im Maschinen- und Anlagenbau sind zumeist noch von einer sequenziellen und disziplinspezifischen Herangehensweise geprägt, die daraus resultiert, dass früher mechanische Lösungen den wesentlichen Anteil an der Wertschöpfung ausmachten, während Elektronik und Software eher als nachgelagerte Randthemen bei einer Maschinen- oder Anlagenentwicklung angesehen wurden [4]. Aufgrund stetig steigender Rechenleistungen von kostengünstigeren Prozessoren ist derzeit allerdings der Trend zu beobachten, dass ein Großteil der Systemfunktionalität softwaretechnisch durch Steuerungsfunktionen realisiert wird (vgl. Abbildung 1). Nach [5] ist Software bereits jetzt in weiten Teilen der Industrie ein wesentlicher Treiber für Innovationen geworden. Diesem Trend müssen zukünftig auch die Entwicklungsprozesse Rechnung tragen. Statt mechanisch geprägter Entwicklungsprinzipien müssen interdisziplinäre Herangehensweisen gefunden werden, die ein synergetisches Zusammenwirken der an der Entwicklung beteiligten Disziplinen Mechanik, Elektronik und Software ermöglichen und so zu einer besseren Produktqualität und zu einer Verkürzung von Entwicklungszeiten beitragen können [6].

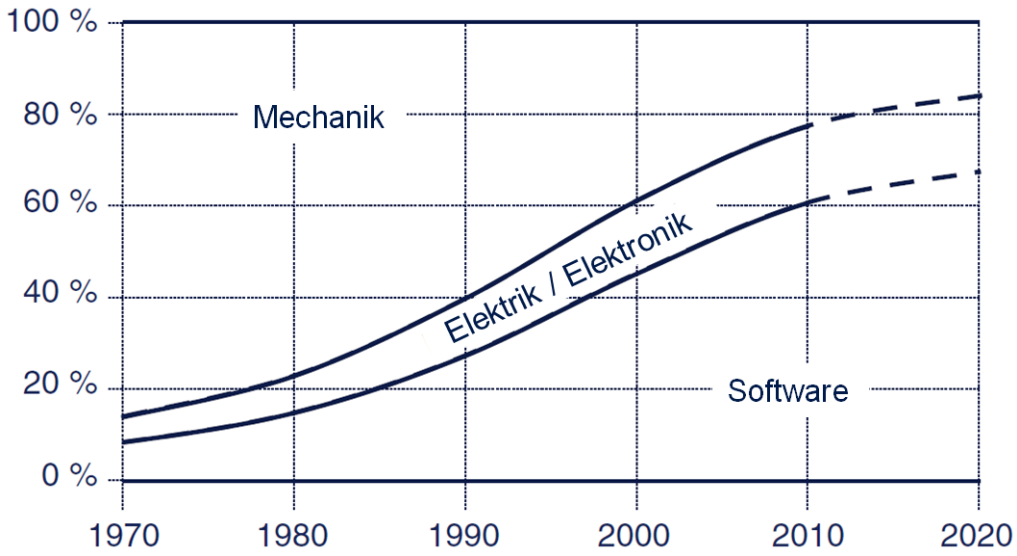


Abbildung 1: Steigende Bedeutung der Software für die Realisierung von Funktionen im Maschinen- und Anlagenbau [7]

Neben der Entwicklung methodischer Vorgehensmodelle (z.B. [8]) bildet die modellbasierte Entwicklung von mechatronischen Systemen einen vielversprechenden Lösungsansatz, um der zunehmenden Interdisziplinarität gerecht zu werden [6]. Modellbasierte Entwicklungsprinzipien haben ihren Ursprung in der Softwareentwicklung. Hierbei werden Modelle vor der eigentlichen Implementierung erstellt, um einerseits das Verständnis für die Funktionalität des zu entwickelnden Programms zu erhöhen und andererseits die Analyse von Softwarelösungen zu vereinfachen [9]. Im Kontext mechatronischer Systeme wird hingegen unter einem modellbasierten Entwicklungsansatz die Ausrichtung des gesamten Entwicklungsprozesses an einem zentralen und interdisziplinären Systemmodell verstanden [10]. Aktuell werden digitale Modelle und Werkzeuge bereits innerhalb einzelner Disziplinen eingesetzt, wie zum Beispiel bei der Konstruktion von Bauteilen in CAD-Werkzeugen, wodurch allerdings Inkonsistenzen zwischen Teillösungen entstehen können [11]. Durch eine frühzeitige Ausrichtung der Entwicklung an einem zentralen Systemmodell, welches von allen Beteiligten aufgebaut wird, kann von Beginn an ein gemeinsames Grundverständnis über das zu entwickelnde System generiert werden. Zudem können Arbeitsschritte und Teilerkenntnisse einfach synchronisiert werden [12].

Dem skizzierten Potenzial einer modellbasierten Entwicklung steht aktuell noch der Aufwand einer zusätzlichen Modellerstellung und -pflege gegenüber, der Unternehmen daran hindert, modellbasierte Entwicklungsansätze in ihren Prozessen einzusetzen [13]. Um dieses Hemmnis aufzulösen, besteht grundsätzlich die Möglichkeit, abgebildete Modellinhalte über den gesamten Entwicklungsverlauf nutzbar zu machen, um so den anfänglichen Modellierungsaufwand durch Einsparungen in späteren Entwicklungsaktivitäten zu rechtfertigen [14]. Nur wenn es zukünftig gelingt, einen modellbasierten Ansatz ganzheitlich in die Entwicklungsprozesse des Maschinen- und Anlagenbaus einzubetten und an dessen spezifischen Anforderungen auszurichten, kann sich dieser zukunftsorientierte und interdisziplinäre Ansatz langfristig innerhalb der Produktionstechnik etablieren.

1.2 Vorarbeiten im Rahmen des Forschungsprojektes AutoVIBN

Vor dem eben dargelegten Hintergrund widmete sich bereits das vorangegangene AiF-Forschungsprojekt AutoVIBN (Automatische Generierung von Verhaltensmodellen aus CAD-Daten für die qualitätsorientierte Virtuelle Inbetriebnahme) der Fragestellung, wie ein zentrales Systemmodell innerhalb der Entwicklungsprozesse des Maschinen- und Anlagenbaus eingesetzt und etabliert werden kann. Neben dieser Fragestellung lag ein zentraler Fokus auf der automatischen Generierung von Verhaltensmodellen für die Simulation, um dadurch eine aufwandsarme Virtuelle Inbetriebnahme (VIBN) einer Maschinensteuerung zu ermöglichen [15].

Diese Fragestellungen wurden in AutoVIBN aufgegriffen und es konnte eine Beschreibungstechnik entwickelt werden, die eine disziplinunabhängige, funktionale Modellierung von mechatronischen Systemen erlaubt. Die erarbeitete Modellierungstechnik basiert auf einer begrenzten Anzahl von Beschreibungsmitteln, welche eine definierte Syntax und Semantik besitzen [16]. Das zentrale Element der Modellierung ist die Komponente, mit deren Hilfe ein mechatronisches System in dezentrale und überschaubare Einheiten gegliedert werden kann. Dadurch, dass mehrere Komponenten zu übergeordneten Komponenten zusammengefasst werden können, können auch umfangreiche Systeme beherrschbar und verständlich gestaltet werden [16]. Um einen Datenaustausch einzelner Komponenten untereinander oder zwischen Hierarchieebenen zu ermöglichen, besitzen Komponenten Ports, die mittels sog. Kanäle verbunden werden können. Abbildung 2 zeigt exemplarisch eine Komponentenstruktur für das einfache Fallbeispiel zweier Förderbänder. Um schließlich funktional zu beschreiben, wie sich ein modelliertes System verhält, können einzelnen Komponenten Verhaltensmuster in Form von hybriden Zustandsautomaten zugewiesen werden. Diese erlauben eine kompakte und systematische Darstellung sowohl des diskreten Verhaltens, was v.a. auf die Steuerung zutrifft, als auch des kontinuierlichen Verhaltens, was hauptsächlich für die Abbildung elektromechanischer Maschinenbestandteile wie Sensoren oder Aktoren genutzt wird. Die hierbei eingesetzten Zustandsgraphen und einfachen Differenzialgleichungen sind für die Entwickler aus allen beteiligten Fachdisziplinen nachvollziehbar und leicht verständlich [16].

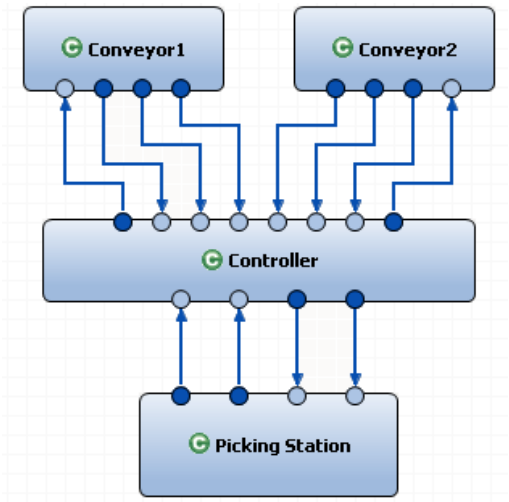


Abbildung 2: Aufbau eines Systems durch Komponenten, Ports und Kanäle [17]

Das eben vorgestellte Komponentenmodell wird durch ein Geometriemodell ergänzt, indem einzelnen Komponenten dreidimensionale Objekte zugeordnet werden [16].

Dadurch können einerseits Platzbedarfe des Systems und Bewegungen visualisiert werden sowie andererseits Kollisionen von Systembestandteilen detektiert werden. Hierzu ist eine stark vereinfachte Beschreibung mittels geometrischer Grundelemente (wie Boxen oder Zylindern) ausreichend, während Detailbeschreibungen bewusst weggelassen werden, um den Fokus nicht allein auf mechanische Entwicklungsaufgaben zu legen. Es können allerdings kinematische Freiheitsgrade von Komponenten im Modell hinterlegt werden, so dass auch räumliche Bewegungsabläufe des Systems erfasst und simuliert werden können [16]. Da ein wesentlicher Anteil der Funktionen von Maschinen und Anlagen den Transport und die Manipulation von Material betrifft, wird das Geometriemodell noch durch ein explizites Materialflussmodell ergänzt. Hierbei werden Materialobjekte von der Simulationsumgebung verwaltet und durch die Komponenten der Anlage beeinflusst. Die Bewegung und Änderung des Materials wird dabei durch Kollisionen und Verhaltensbeschreibungen in den Automaten ermittelt [16].

Neben der Nutzung des Funktionsmodells als zentrales Kommunikationsmedium zwischen den beteiligten Entwicklungsdisziplinen kann dieses auch verwendet werden, um automatisch Verhaltensmodelle zu generieren, die für eine VIBN der Anlagensteuerung benötigt werden. Für die automatische Modellgenerierung wird das abstrakte Modell der Maschine in der Funktionsbeschreibung mit Details aus den CAD-Systemen angereichert (vgl. Abbildung 3). Die entsprechenden Protokolle sind in einer Bibliothek abgelegt, die während der Generierung eingebunden wird. Durch das Zusammenwirken von Funktionsbeschreibung, CAD-Daten und der Bibliothek kann ein Maschinenmodell als C++-Programm abgeleitet werden. Dieses Modell läuft auf einem Simulationsrechner, der entweder mit der realen Steuerung über einen Feldbus oder mit einer virtuellen Steuerung verbunden wird. Das Maschinenverhalten wird auf dem C++-Simulator berechnet und ist über eine Bedienerschnittstelle beeinflussbar. Die Simulationsergebnisse werden dem Busprotokoll entsprechend in Steuerungseingänge übersetzt und auf die Karte geschrieben. Die Steuerungsausgangssignale werden von der Simulationskarte gelesen und in Eingangssignale für die Simulation umgewandelt. Für die Nachvollziehbarkeit der Maschinenabläufe und -bewegungen ist eine Visualisierung unabdingbar. Hierzu werden entsprechende Modelle aus dem Mechanik-CAD-System ausgeleitet und angepasst. Die Kopplung der Visualisierung erfolgt an den Simulationskern, der die Positions- und Zustandswerte übergibt [18].

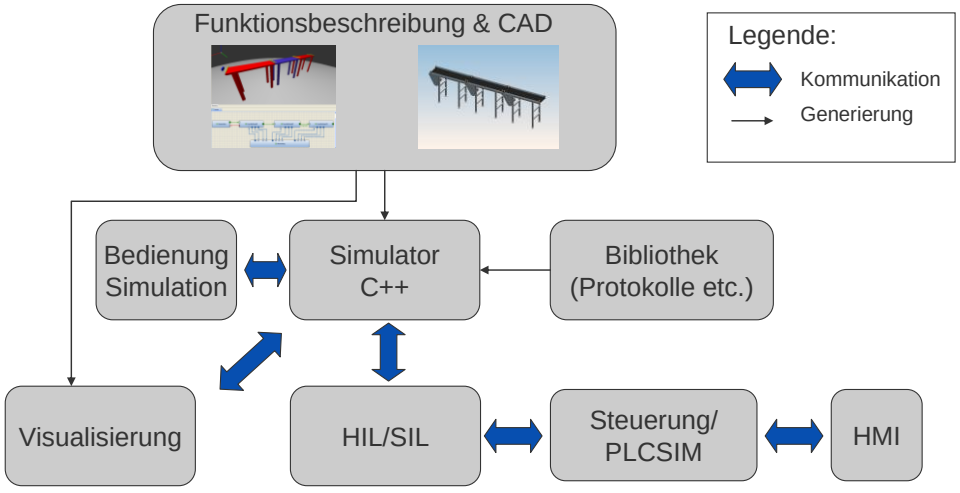


Abbildung 3: Prinzipieller Aufbau der VIBN-Simulationsumgebung [18]

Ausgehend von den oben genannten Arbeiten zum Aufbau des Funktionsmodells und zur Realisierung des Simulators für die Virtuelle Inbetriebnahme wurde im Rahmen von AutoVIBN zudem eine Methode erarbeitet, die eine interdisziplinäre Entwicklung von mechatronischen Systemen unterstützt und die Modellerstellung für die VIBN von Maschinen und Anlagen weitestgehend automatisiert. Hierfür wurden Daten (z.B. aus dem CAD) genutzt, die im Entwicklungsprozess bereits vorhanden sind. Die entwickelten Techniken und Methoden wurden schließlich an zwei industriellen Fallstudien erfolgreich erprobt. [19][20][21]

1.3 Zielstellung des Forschungsprojektes IMoMeSA

Basierend auf den Ergebnissen aus AutoVIBN war das Ziel des Forschungsprojektes IMoMeSA die Definition eines erweiterten integrierten Funktionsmodells, das entsprechende Verarbeitungsschritte, wie zum Beispiel die Generierung, die Synchronisation und die Konsistenzprüfung weiterer Modelle, ermöglicht. Hierdurch soll der Entwicklungsprozess im Maschinen- und Anlagenbau nachhaltig unterstützt werden. Die Intention ist dabei zum einen die Beschleunigung der Entwicklung durch eine Automatisierung von Tätigkeiten. Zum anderen soll aber auch eine Steigerung der Qualität durch eine Reduktion fehleranfälliger, manueller Schritte und den Einsatz systematischer Analysetechniken realisiert werden, die durch eine Teilautomation erst ökonomisch sinnvoll einsetzbar sind.

Zur Realisierung dieser Zielstellung gliederte sich das Projekt IMoMeSA in fünf inhaltliche Schwerpunktthemen:

1.) Die Definition **formaler Anforderungen** erlaubt eine präzise und kompakte Beschreibung der Anforderungen an eine Maschine oder Anlage. Eine modellbasierte Notation gestattet die Prüfung von Konsistenzproblemen und die Herstellung von Bezügen zu anderen Entwicklungsartefakten, die mit einer textuellen Beschreibung nicht möglich wäre.
2.) Die **Generierung von Steuerungscode** aus einem Funktionsmodell erhöht die Entwicklungsgeschwindigkeit, da die Steuerungsfunktionen in einem abstrakten Modell schneller beschrieben werden können. Zudem lässt sich die Fehlerbehandlung, die einen wesentlichen Teil des Programmcodes einnimmt, durch eine Angabe von generischen Fehlerbehandlungsroutinen oftmals vollständig ableiten.
3.) Die **Zusammenarbeit mit bestehenden Werkzeugketten** umfasst den Datenaustausch mit anderen Entwicklungswerkzeugen, aber auch die teilweise oder vollständige Generierung von (Rahmen-)Modellen für MCAD- und ECAD-Systeme, für die Mehrkörpersimulation (MKS) oder für eine FE-Simulation.
4.) Eine **Fehlermöglichkeits- und einflussanalyse** (FMEA) ist eine Analysetechnik zur Vermeidung von Systemfehlern und zur Abschätzung und Eindämmung von deren Auswirkungen, die in vielen Industriebereichen eingesetzt oder sogar vorgeschrieben wird. Durch die Verwendung von geeigneten Modellen lässt sich zum einen der Aufwand für eine FMEA deutlich reduzieren und zum anderen die Genauigkeit der Analyse verbessern.
5.) Eine **Entwicklungsmethodik** ermöglicht die Abstimmung und Verschmelzung der angestrebten Projektergebnisse. Dabei soll untersucht werden, wie ein idealisierter modellbasierter Entwicklungsprozess unter Berücksichtigung der Ergebnisse aus AutoVIBN und IMoMeSA aussehen könnte.

Die Grundlagen der Funktionsbeschreibung und die Virtuelle Inbetriebnahme wurden bereits im Projekt AutoVIBN ausführlich untersucht. Diese Ergebnisse wurden im Rahmen von IMoMeSA mit neuen Ansätzen verbunden. Das Gesamtkonzept des modellbasierten Entwicklungsansatzes ist in Abbildung 4 dargestellt, wobei sich die Betätigungsfelder des Projektes AutoVIBN (blau) farblich von den neuen inhaltlichen Schwerpunkten (grün) unterscheiden.

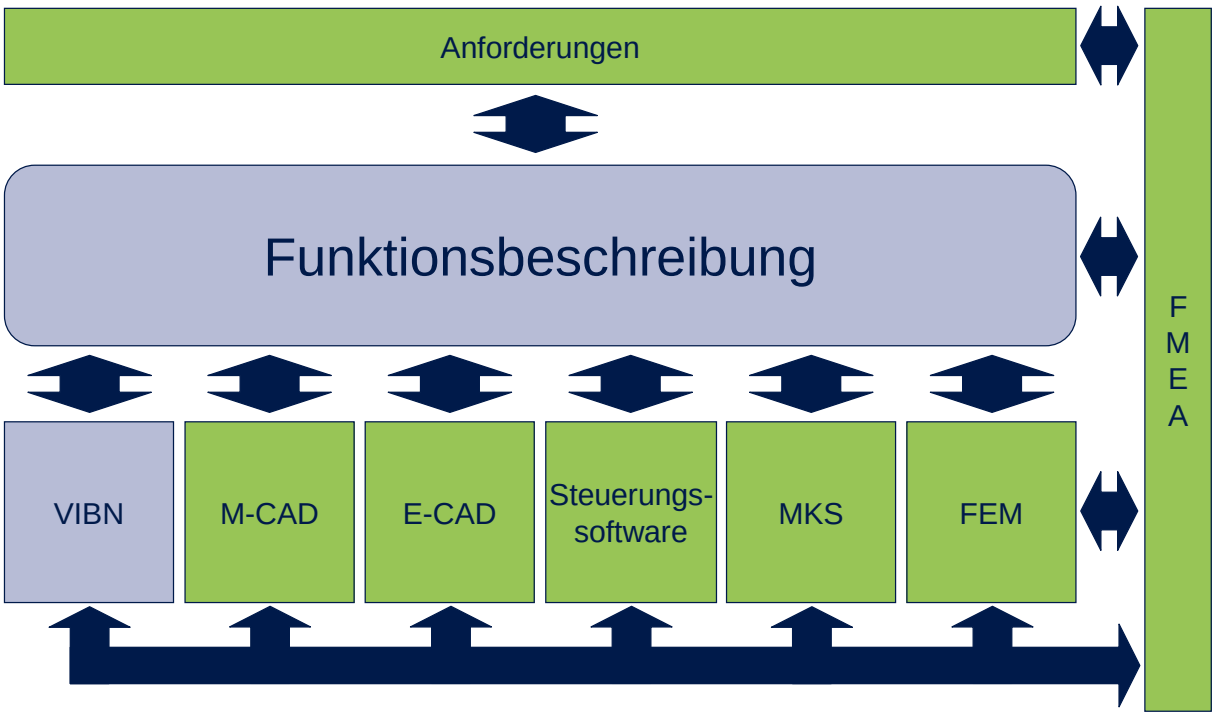


Abbildung 4: Überblick der inhaltlichen Schwerpunkte des Forschungsprojektes IMoMeSA

Neben den inhaltlichen Schwerpunkten widmete sich das Forschungsprojekt IMoMeSA ebenfalls der Aufgabe, die Erkenntnisse bestmöglich an den derzeitigen Herausforderungen des Werkzeugmaschinenbaus auszurichten. Zu diesem Zweck sollten einerseits die Anforderungen der beteiligten Projektpartner an das Projekt erfasst und der aktuelle Stand der Technik hinsichtlich des Einsatzes von Modellen ermittelt werden. Andererseits sollten die entwickelten Methoden und Techniken kontinuierlich an praxisnahen Fallstudien evaluiert werden, um die Übertragbarkeit der Ergebnisse und Schlussfolgerungen zu ermöglichen.

1.4 Aufbau des Abschlussberichtes

Um darzustellen, welche Ergebnisse innerhalb der adressierten Schwerpunkte des Forschungsprojektes IMoMeSA erarbeitet werden konnten, gliedert sich dieser Abschlussbericht wie folgt (vgl. Abbildung 5):

Zunächst werden in **Kapitel 2** die durchgeführten Arbeiten im Bereich der Anforderungsanalyse zusammengefasst. Dabei wird zunächst die Vorgehensweise vorgestellt, mit der unter Einbeziehung des projektbegleitenden Ausschusses die Anforderungen der Industrie an eine modellbasierte Entwicklungsmethodik aufgenommen wurden. Anschließend werden die Ergebnisse der Analyse dargelegt, bevor schließlich die daraus abgeleiteten Schlussfolgerungen für das Projekt IMoMeSA präsentiert werden.

Aufbauend auf den Erkenntnissen der Anforderungsanalyse wird in **Kapitel 3** die erarbeitete modellbasierte Entwicklungsmethodik vorgestellt, in welche die adressierten Arbeitsschwerpunkte eingebettet sind. Die Entwicklungsmethodik unterscheidet dabei auf oberster Ebene grundsätzlich eine Konzeptions- und eine Verfeinerungsphase, was sich auch im Aufbau dieses Kapitels widerspiegelt. Im Bereich der Konzeption werden insbesondere die Schwerpunktthemen des Anforderungs- und Fehlermanagements adressiert, während die Codegenerierung und die Zusammenarbeit mit digitalen Werkzeugen in der Verfeinerung beschrieben werden.

In **Kapitel 4** des Abschlussberichtes wird die erarbeitete modellbasierte Entwicklungsmethodik schließlich anhand eines praxisnahen Fallbeispiels evaluiert. Dabei werden zunächst das Fallbeispiel sowie die Vorgehensweise bei der Evaluation beschrieben, bevor die Ergebnisse der Anwendung der Entwicklungsmethodik an dem gewählten Beispiel, einem Bedruckungsmodul einer miniaturisierten Verpackungsanlage, vorgestellt werden. Abschließend werden diese Ergebnisse bewertet, um die Stärken und Schwächen des erarbeiteten Ansatzes insbesondere hinsichtlich einer praktischen Anwendbarkeit in der Industrie besser herausstellen zu können.

In **Kapitel 5** werden schließlich die zentralen Ergebnisse aus dem IMoMeSA-Forschungsprojekt zusammengefasst. Darüber hinaus wird ein Ausblick dargelegt, wie aufbauend auf den Ergebnissen eine langfristige Etablierung der modellbasierten Entwicklungsprinzipien im Maschinen- und Anlagenbau erreicht werden kann.

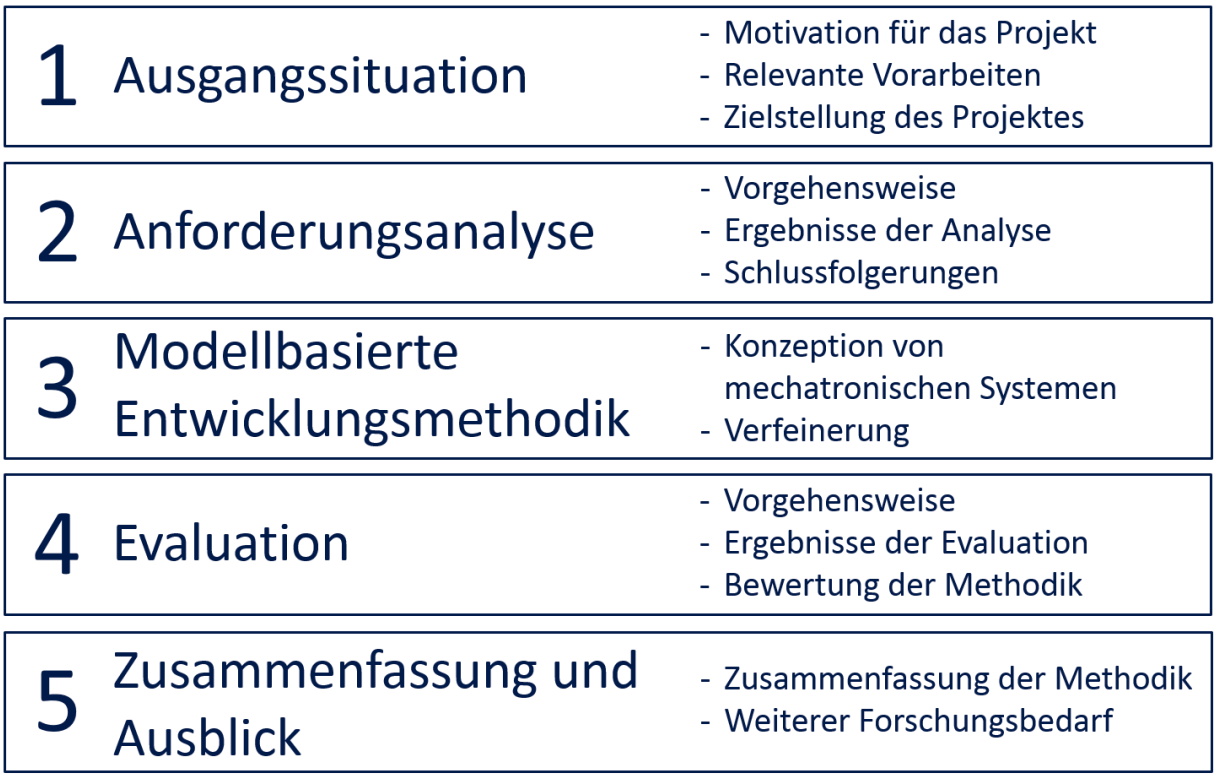


Abbildung 5: Aufbau des Abschlussberichtes

2 Anforderungsanalyse

Neben dem aktuellen Stand der Wissenschaft war es für das Forschungsprojekt IMoMeSA besonders wichtig, die derzeitige Situation bei den beteiligten Projektpartnern zu erfassen. Hierfür wurde durch verschiedene Techniken (z. B. Workshops, Interviews, etc.) unter anderem aufgenommen, wie Entwicklungsprozesse im Werkzeugmaschinenbau aktuell ausgeprägt sind und welche digitalen Modelle sowie diesbezügliche Werkzeuge dabei zum Einsatz kommen.

Abschnitt 2.1 gibt zunächst einen detaillierten Einblick in die einzelnen Maßnahmen, die im Rahmen von IMoMeSA durchgeführt wurden, um die aktuelle Situation der Projektpartner aufzunehmen und dadurch im Verlauf des Projektes zu berücksichtigen. Anschließend werden in Abschnitt 2.2 die einzelnen Ergebnisse der Anforderungsanalyse vorgestellt und den Teilbereichen *Entwicklungsprozess*, *digitale Werkzeuge* sowie *Anforderungs- und Fehlerübersicht* zugeordnet. Im letzten Abschnitt (Abschnitt 2.3) wird schließlich dargelegt, welche Schlussfolgerungen aus diesen Ergebnissen gezogen werden konnten und wie diese die inhaltlichen Arbeiten im Rahmen von IMoMeSA beeinflussten.

2.1 Vorgehensweise

Um einen ersten Überblick über die aktuelle Situation in den Unternehmen des projektbegleitenden Ausschuss zu erhalten, wurden zu Beginn des Forschungsprojektes Interviews bei den Unternehmen *Liebherr*, *Index Werke* und *Bihler* durchgeführt. Ziel dieser ersten Analyse war es, den grundsätzlichen Ablauf von Entwicklungsprozessen in den jeweiligen Unternehmen kennenzulernen sowie die digitalen Werkzeuge aufzunehmen, die innerhalb dieser Prozesse zum Einsatz kommen. Ein besonderer Fokus lag bereits bei dieser ersten Maßnahme darauf, mögliche Schwachstellen im Prozess bzw. bei den digitalen Werkzeugen zu identifizieren, die mittels eines modellbasierten Entwicklungsansatzes aufgehoben werden können. Die Ergebnisse dieser drei Einzelinterviews wurden anschließend synthetisiert und im ersten Statustreffen des projektbegleitenden Ausschuss diskutiert. Dadurch konnte zwar zum einen die Richtigkeit der bisherigen Analyseergebnisse bestätigt werden, allerdings wurde auch offensichtlich, dass insbesondere in den Bereichen Entwicklungsprozess sowie Anforderungs- und Fehlermanagement noch aufbauende Analysearbeiten notwendig sind, um eine zielgerichtete Arbeit in den einzelnen Arbeitspaketen des IMoMeSA-Projektes zu ermöglichen.

Aus diesem Grund wurde unter Beteiligung des gesamten projektbegleitenden Ausschusses ein Workshop bei der Firma *Schütte* in Köln durchgeführt. Ziel dieses Workshops war die Aufnahme eines generischen Entwicklungsprozesses, der den aktuellen Stand im Werkzeugmaschinenbau, auch mit seinen derzeitigen Schwachstellen, bestmöglich abbildet. Darüber hinaus wurden typische Anforderungen und Verhaltensfehler, die bei der Entwicklung von Werkzeugmaschinen berücksichtigt werden müssen, aufgenommen. Zudem konnte eine erste Klassifikation dieser Anforderungen und Verhaltensfehler erarbeitet werden. Nach der Verdichtung dieser Erkenntnisse wurden der erarbeitete Entwicklungsprozess sowie die Anforderungs- und Fehlerklassifikation dem projektbegleitenden Ausschuss im zweiten Statustreffen vorgestellt, diskutiert und anschließend finalisiert. Sowohl die Anforderungs- als auch die Fehlerklassifikation sollten im Laufe der Projektes dazu

dienen, die Arbeiten im Bereich der Anforderungsformalisierung und der FMEA an den tatsächlichen Gegebenheiten des Werkzeugmaschinenbaus auszurichten.

2.2 Ergebnisse der Analyse

In den nachfolgenden Teilabschnitten werden die Erkenntnisse aus dem eben beschriebenen, zweistufigen Analyseverfahren zusammengefasst. Zunächst wird ein *generischer Entwicklungsprozess* im Werkzeugmaschinenbau vorgestellt, der so weit abstrahiert wurde, dass er den aktuellen Stand aller an der Analyse beteiligten Unternehmen abbildet und dadurch eine gewisse Allgemeingültigkeit besitzt. Darauf aufbauend wird skizziert, welche *digitalen Werkzeuge* innerhalb dieses Prozesses aktuell zum Einsatz kommen, um so Aussagen zu ermöglichen, inwiefern die Zusammenarbeit eines zentralen Funktionsmodells mit etablierten Werkzeugketten im Werkzeugmaschinenbau sinnvoll erfolgen kann. Schließlich werden die *Anforderungs- und Fehlerübersichten bzw. -klassifikationen* vorgestellt, die es ermöglichen, das gesamte Spektrum an Anforderungen und Verhaltensfehlern von Werkzeugmaschinen abzubilden. Dadurch können die Arbeiten zur Anforderungsformalisierung und zur FMEA kontinuierlich an diesen Übersichten gespiegelt werden und dadurch ebenfalls eine gewisse Allgemeingültigkeit erlangen.

2.2.1 Generischer Entwicklungsprozess mit aktuellen Hemmnissen

Obwohl die Bandbreite der Unternehmen des projektbegleitenden Ausschusses von der Einzel- bis zur Serienfertigung reicht, konnten bei geeigneter Abstraktion große Gemeinsamkeiten in den Entwicklungsprozessen identifiziert werden. Insbesondere konnte bei der Abfolge einzelner Phasen auf oberster Ebene eine grundsätzliche Übereinstimmung identifiziert werden (vgl. Abbildung 6).

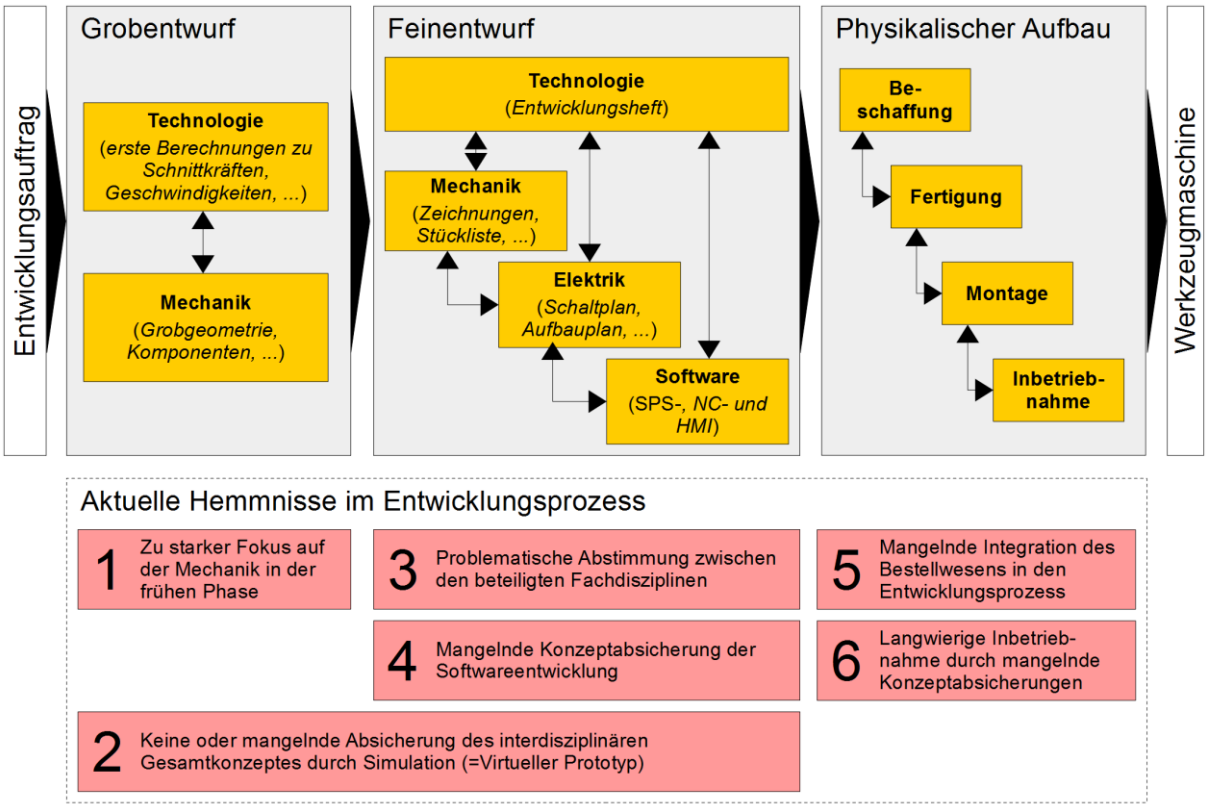


Abbildung 6: Generischer Entwicklungsprozess mit aktuellen Hemmnissen

Den Ausgangspunkt eines typischen Entwicklungsprozesses bildet ein Entwicklungsauftrag, der sich aus eigenen Marktsondierungen, technologischen Weiterentwicklungen oder aus unmittelbaren Kundenaufträgen ergeben kann. Auf dieser Basis werden im Rahmen des Grobentwurfs erste mechanische Konzepte erarbeitet und gegeneinander abgewogen. Zur Konzeptabsicherung werden bereits in dieser Phase von der Technologieabteilung erste Berechnungen zu Zerspankräften oder Verfahrgeschwindigkeiten durchgeführt, um so die grundsätzliche Realisierbarkeit der mechanischen Lösung zu überprüfen. Nach erfolgter Freigabe des Grobkonzeptes wird dieses im Feinentwurf in den beteiligten Fachdisziplinen der Mechanik, Elektrik und Software weiter ausdetailliert, wobei ein eher sequenzielles Vorgehen charakteristisch ist. Dabei erarbeitet zunächst die Mechanik detaillierte CAD-Zeichnungen des zu entwickelnden Systems, aus denen schließlich technische Zeichnungen und Stücklisten für den physikalischen Aufbau abgeleitet werden können. Das detaillierte Mechanik-Konzept wird anschließend von der Elektrik-Abteilung verwendet, um den Schalt- oder den Aufbauplan auszuarbeiten. Auf dieser Basis wiederum können in der Software-Abteilung die Maschinensteuerung (SPS und NC) sowie die Benutzerschnittstelle (HMI) implementiert werden. Während dieser Aktivitäten wird in der Technologieabteilung ein sog. Entwicklungsheft erstellt, in das alle wichtigen Daten aus den einzelnen Disziplinen eingepflegt werden. Nach Abschluss des Feinentwurfs kann mit dem physikalischen Aufbau der Werkzeugmaschine fortgefahren werden. Diese Phase beinhaltet zunächst die Beschaffung von Zukaufteilen sowie die Herstellung eigens zu fertigender Bauteile. Diese können anschließend montiert und somit zu einer Werkzeugmaschine zusammengesetzt werden. In einem letzten Schritt wird die Software auf die Steuerung aufgespielt, wodurch schließlich die Werkzeugmaschine unter realen Einsatzbedingungen in Betrieb genommen werden kann.

Auf Basis dieser Prozessbeschreibung konnten bereits erste Hemmnisse identifiziert werden, die in einzelnen Phasen einen reibungslosen Ablauf des Entwicklungsprozesses behindern. Unter diesen ist zunächst die zu starke Fokussierung der Mechanik in der frühen Entwicklungsphase bzw. im Grobentwurf zu nennen. Trotz einer steigenden Bedeutung von Elektrik und Software ist die Mechanik nach wie vor alleine für die Festlegung eines grundsätzlichen Maschinenkonzeptes verantwortlich. Die elektrische oder steuerungstechnische Machbarkeit wird in dieser Phase nicht überprüft, was sich später in kostspieligen Iterationsschleifen niederschlagen kann. Eng verknüpft mit diesem Punkt ist die mangelnde simulative Absicherung eines interdisziplinären Gesamtkonzepts, die sich sowohl auf die Phase des Grob- als auch des Feinentwurfs bezieht. Während einzelne Simulationsmethoden bereits innerhalb spezifischer Disziplinen etabliert sind, bedarf es mittlerweile auch der Erprobung an einem interdisziplinären virtuellen Prototyp, um bereits vor dem physikalischen Aufbau das Zusammenspiel einzelner Teillösungen evaluieren zu können. Des Weiteren tritt in der Phase des Feinentwurfs das Problem einer grundsätzlich problematischen Abstimmung zwischen den beteiligten Fachdisziplinen auf. Die Weiterleitung von Informationen erfolgt zumeist nur auf Basis von Dokumenten wie Stücklisten oder Schaltplänen, wohingegen eine interdisziplinäre Lösungsfindung nur sehr selten stattfindet. Der sequenzielle Charakter der Entwicklungsprozesse verstärkt dieses Problem insofern, als dass offene Schwierigkeiten oftmals einfach an nachgelagerte Disziplinen weitergegeben werden, ohne dass ein interdisziplinärer Austausch

stattgefunden hat. Die Konsequenzen fallen schlussendlich zumeist der Software-Abteilung zur Last, die als letztes Glied in der Kette eine Vielzahl der verbliebenen Probleme beheben muss oder eine Überarbeitung durch andere Disziplinen anstoßen muss. Da allerdings in dieser Phase noch keine realen Aufbauten existieren, an denen die Software getestet werden kann, können die erarbeiteten Lösungen nur unzureichend abgesichert werden. Erst bei der Inbetriebnahme, wenn die Software auf die Steuerung der Werkzeugmaschine aufgespielt wird, kann die Steuerungs-funktionalität angemessen abgesichert werden. Aufgrund des großen Umfangs, den Steuerungsprogramme mittlerweile einnehmen, ist dieser Schritt oftmals sehr langwierig und kostspielig und führt insgesamt zu Problemen hinsichtlich des Auslieferungszeitpunktes. Ein weiteres Problem, das im Rahmen der Anforderungs-analyse identifiziert werden konnte, liegt in der mangelnden Integration des Bestellwesens in den Entwicklungsprozess. Oftmals werden für den Aufbau der Werkzeugmaschine bestimmte Komponenten benötigt, die eine lange Lieferzeit haben. Da der Bestellvorgang allerdings zumeist erst nach abgeschlossener Konstruktion erfolgt, kommt es zu Stillständen in der Montage, in denen auf das Eintreffen von Lieferteilen gewartet wird. Wenn es gelingt, diese Problematik bereits in der Konstruktion besser zu berücksichtigen, könnten solche Engpässe durch eine frühzeitige Auslösung von Bestellungen vermieden werden.

2.2.2 Einsatz digitaler Werkzeuge während des Entwicklungsprozesses

Bei den Unternehmen des projektbegleitenden Ausschusses ist der Einsatz digitaler Modelle und Werkzeuge am weitesten in der *Konstruktion* verbreitet. Zur mechanischen Konstruktion von Bauteilen und -gruppen kommen aktuell Werkzeuge von vielen verschiedenen Herstellern zum Einsatz. Unter diesen sind exemplarisch NX der Fa. Siemens, Creo der Fa. PTC und SolidWorks der Fa. Dassault Systèmes zu nennen, die im Werkzeugmaschinenbau derzeit eine breite Anwendung finden. Im Bereich der elektrischen Konstruktion konzentriert sich dagegen der Einsatz digitaler Werkzeuge zur Ausarbeitung von Schaltplänen oder Schaltschränken zunehmend auf die Software EPLAN P8 der Fa. EPLAN. Andere Werkzeuge wie E³.series der Fa. Zuken oder ELCAD der Fa. Autotec finden nur noch vereinzelt Anwendung. Ähnlich verhält es sich im Bereich der Steuerungsentwicklung. Sowohl im SPS- wie auch im NC-Bereich kommen zumeist Produkte der Fa. Siemens zum Einsatz (Simatic, Sinumerik). Andere Hersteller wie bspw. die Fa. B&R oder die Fa. Beckhoff bieten zwar grundsätzlich vergleichbare Produkte an, die aber nur selten zur Anwendung kommen, da die Siemens-Produkte oft sogar von den Kunden des Werkzeugmaschinenbaus gefordert werden. In allen genannten Bereichen ist der Einsatz digitaler Werkzeuge bereits seit langem etabliert. Die einzelnen Werkzeuge werden von den Unternehmen beherrscht, Probleme treten allerdings oftmals an den Schnittstellen zwischen Werkzeugen auf, an denen Daten oft nur über Umwege oder teilweise überhaupt nicht ausgetauscht werden können.

Ein weiterer Bereich innerhalb der Entwicklung, in dem vermehrt digitale Werkzeuge zum Einsatz kommen, ist die *Simulation*. Für Spezialaufgaben wie bspw. die Simulation von Bewegungsabläufen mittels MKS oder die Analyse von Steifigkeiten mittels FEM haben sich eigene Werkzeuge wie MotionSolve der Fa. Altair, Matlab/Simulink der Fa. MathWorks oder ANSYS der Fa. Ansys etabliert. Für spezielle Anwendungen werden diese Werkzeuge oft sogar unternehmensspezifisch erweitert,

da die Standardumfänge dieser Tools nicht den spezifischen Anforderungen gerecht werden. Während sich die Simulation in diesen Teilbereichen bereits etabliert hat, mangelt es derzeit noch an dem Einsatz übergreifender Simulationsansätze, die das ganzheitliche Maschinenverhalten aus der Sicht aller an der Entwicklung beteiligter Disziplinen abbilden. Eine erste Möglichkeit hierfür bietet die Virtuelle Inbetriebnahme, mit der implementierte Maschinensteuerungen mit einem Verhaltensmodell der Werkzeugmaschine in Betrieb genommen werden können. Exemplarische Werkzeuge, wie bspw. die Software WinMOD der Fa. Mewes & Partner oder die Software Virtuos der Fa. ISG, kommen bereits vereinzelt zum Einsatz, allerdings verhindert der Aufwand einer zusätzlichen Modellerstellung derzeit noch eine breite Anwendung.

Neben der Konstruktion und Simulation finden digitale Werkzeuge auch in *weiteren Bereichen* des Entwicklungsprozesses Anwendung. Unter diesen sind zunächst ERP-Werkzeuge zu nennen, die zur Verwaltung von Stücklisten und zum Auslösen von Bestellungen verwendet werden. Dabei konzentriert sich der Markt aktuell auf wenige Anbieter, unter denen vor allem die Fa. SAP zu nennen ist. Weitere Bereiche, die bei den Unternehmen des Werkzeugmaschinenbaus bereits durch digitale Werkzeuge unterstützt werden, sind die Auftragsabwicklung und das Kundenmanagement. Oftmals kommen in diesen Bereichen ebenfalls die o.g. ERP-Systeme zum Einsatz, aber auch Speziallösungen, wie bspw. von der Fa. Camos, finden vereinzelt Anwendung. Abschließend seien noch die Entwicklungsbereiche des Anforderungs- und Fehlermanagements genannt, in denen ebenfalls digitale Werkzeuge zur Aufnahme und Verwaltung von Anforderungen bzw. Verhaltensfehlern eingesetzt werden. Während im Automotive- oder Avionik-Bereich zumeist spezielle Werkzeuge, wie bspw. die Software DOORS der Fa. IBM genutzt werden, ist das Anforderungs- und Fehlermanagement im Werkzeugmaschinenbau zumeist noch Excel- oder Word-basiert. Durch steigende Kundenansprüche sowie immer komplexer werdende Entwicklungsaufgaben genügen diese einfachen Lösungen allerdings oft nicht mehr den aktuellen Gegebenheiten, so dass auch im Werkzeugmaschinenbau zukünftig verstärkt spezielle Tools zum Anforderungs- und Fehlermanagement anzutreffen sein werden.

2.2.3 Anforderungsübersicht und -klassifikation

Während eines typischen Entwicklungsprozesses im Werkzeugmaschinenbau müssen Entwickler mit einer Vielzahl an Anforderungen von unterschiedlichen Autoritäten (=Stakeholdern) umgehen können. Um die Aufgabe des Anforderungsmanagements mittels eines modellbasierten Ansatzes bestmöglich unterstützen zu können, wurden im Rahmen der Anforderungsanalyse typische Anforderungen an Werkzeugmaschinen gesammelt und klassifiziert. Abbildung 7 zeigt die erarbeitete Klassifikation und gibt Beispiele für typische Anforderungen innerhalb der einzelnen Kategorien.

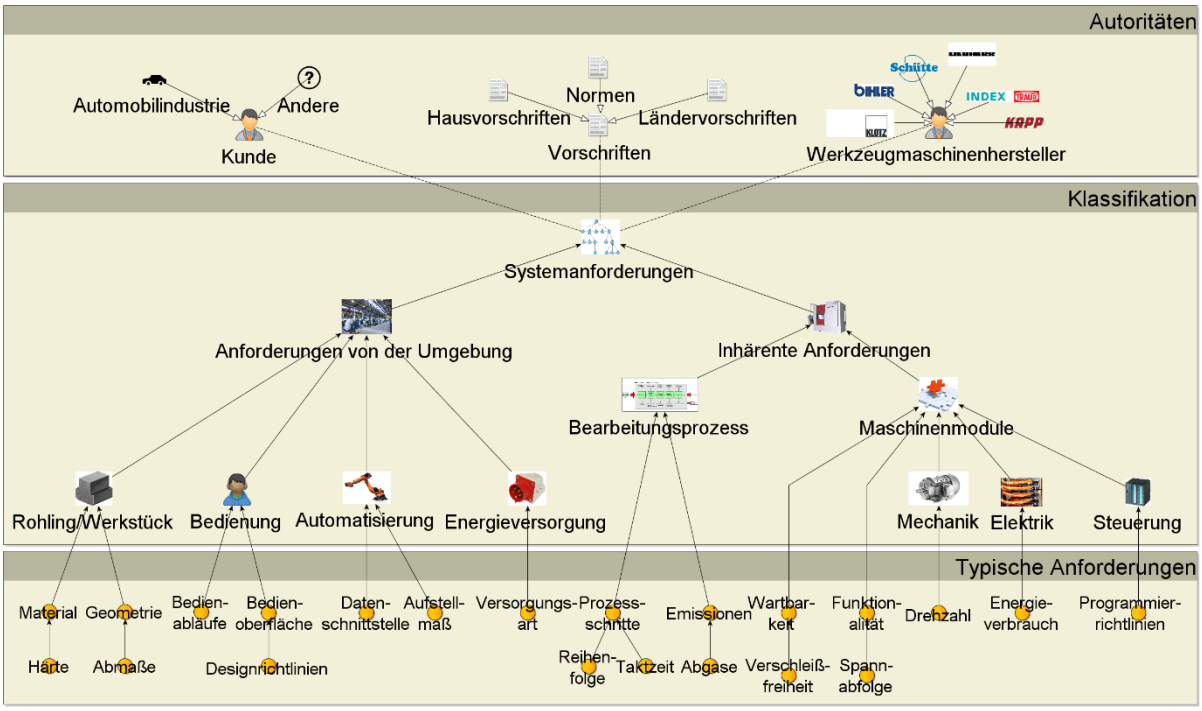


Abbildung 7: Anforderungsklassifikation im Werkzeugmaschinenbau

Der obere Abschnitt von Abbildung 7 gibt zunächst einen Überblick über die einzelnen Autoritäten, die Anforderungen an eine zu entwickelnde Werkzeugmaschine stellen. Unter diesen sind zunächst die Kunden des Maschinenbaus zu nennen, die als spätere Abnehmer in der Regel die zentralen Anforderungen zum Beispiel hinsichtlich Taktzeiten oder Energieverbrauch vorgeben. Darüber hinaus müssen Entwickler aber auch Anforderungen berücksichtigen, die sich aus diversen Vorschriften wie einzuhaltenden Normen oder länderspezifischen Richtlinien ergeben. Schließlich sind auch die Werkzeugmaschinenhersteller selbst als eigene Quelle von Anforderungen zu nennen. Dies kann zunächst dann der Fall sein, wenn ein Entwicklungsauftrag nicht durch einen externen Kunden, sondern durch interne Marktsondierungen angestoßen wird. Aber auch im Falle einer Kundenentwicklung werden vom Werkzeugmaschinen-hersteller typischerweise Vorgaben hinsichtlich der maximalen Kosten oder der Einhaltung von hausinternen Standards gestellt.

Die Anforderungen, die von diesen Autoritäten an das System „Werkzeugmaschine“ gestellt werden, lassen sich grundsätzlich in Anforderungen von der Umgebung und inhärente Anforderungen unterteilen (vgl. Abbildung 7, Mitte). Die erstgenannte Klasse lässt sich anschließend weiter unterteilen, unter anderem in Anforderungen hinsichtlich des Rohlings bzw. des Werkstücks. Diese sind in der Regel eine zentrale Quelle für Anforderungen, da hier typischerweise definiert wird, welche Produkte (Material, Abmaße, …) von der Werkzeugmaschine hergestellt werden müssen bzw. welche Rohmaterialien dafür zum Einsatz kommen sollen. Auch die Maschinenbedienung ist ein wichtiger Faktor für auftretende Anforderungen in der Entwicklung, insbesondere hinsichtlich der zu entwickelnden Mensch-Maschine-Schnittstelle. Oftmals möchten Kunden einheitliche Designrichtlinien in ihrem Maschinenpark verwenden oder geben direkt den grundsätzlichen Ablauf von Benutzerdialogen vor. Darüber hinaus ergeben sich weitere zentrale Anforderungen aus der Automatisierungstechnik, welche die Werkzeugmaschine an ihrem Einsatzort umgibt. Als Beispiele seien an dieser Stelle

Anforderungen hinsichtlich der zu verwendenden Datenschnittstelle oder der genauen Positionierung der Werkzeugmaschine bzw. ihrer Materialflussschnittstellen genannt. Schließlich können Anforderungen von der Umgebung auch aus der Energieversorgung der Werkzeugmaschine resultieren. Ein Beispiel in diesem Bereich ist die Art der Energieversorgung, die sich oft auch aus länderspezifischen Richtlinien ergibt.

Die Anforderungen, welche die Werkzeugmaschine selbst betreffen, lassen sich zunächst unterteilen in Anforderungen, die den zu realisierenden Bearbeitungsprozess beschreiben sowie technische Anforderungen an einzelne Maschinenmodule. Hinsichtlich des Bearbeitungsprozesses ergeben sich typischerweise Anforderungen an die Reihenfolge oder die Taktzeiten von einzelnen Prozessschritten. Darüber hinaus werden oft auch Grenzwerte für Emissionen definiert, die während eines gewissen Betriebszeitraums nicht überschritten werden dürfen. Hinsichtlich einzelner Maschinenmodule werden Anforderungen zumeist an die Wartbarkeit oder die Funktionalität gestellt. Eine Vielzahl der Modulanforderungen lässt sich allerdings weiter in mechanische, elektrische und steuerungstechnische Anforderungen eingruppieren. Hinsichtlich der Mechanik werden beispielsweise Anforderungen an zu realisierende Drehzahlen einzelner Motoren gestellt, bei der Elektrik werden typischerweise Vorgaben bezüglich des Energieverbrauchs von Komponenten getroffen. Hinsichtlich der Steuerungstechnik werden schließlich Anforderungen an einzusetzende Komponenten (SPSen, etc.) gestellt, aber auch kundenspezifische Programmierrichtlinien müssen oftmals berücksichtigt werden.

2.2.4 Fehlerübersicht und -klassifikation

Um Entwicklungstätigkeiten im Bereich der Fehleranalyse und -behebung modellbasiert zu unterstützen, wurden analog zur Anforderungsklassifikation typische Fehler aufgenommen, die während des Betriebs dazu führen können, dass eine Werkzeugmaschine nicht wie gewünscht funktioniert. Abbildung 8 zeigt Beispiele typischer Fehler von Werkzeugmaschinen sowie die abgeleitete Fehlerklassifikation.

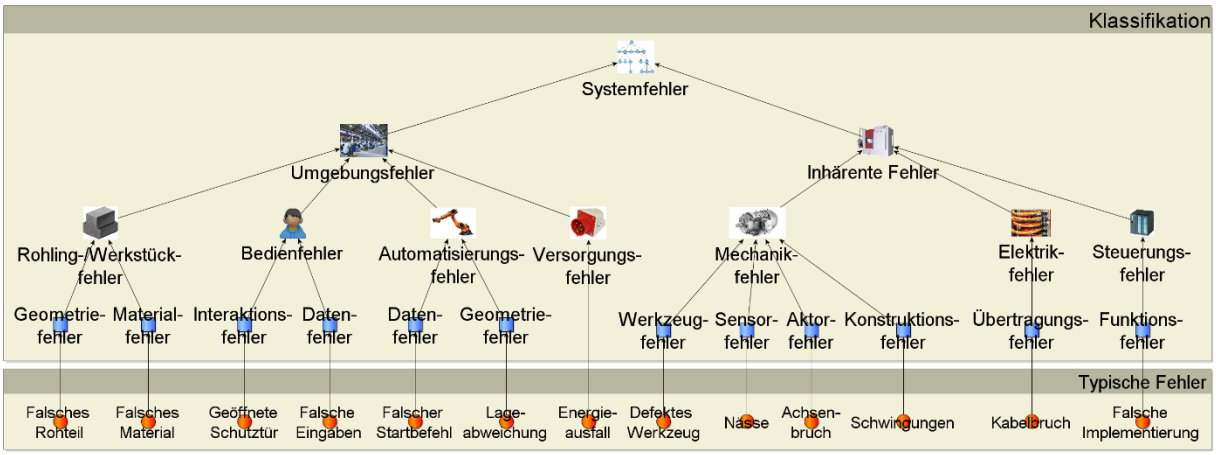


Abbildung 8: Fehlerklassifikation im Werkzeugmaschinenbau

Analog zur Anforderungsklassifikation können Fehler des Systems „Werkzeugmaschine“ in Umgebungsfehler und inhärente Fehler eingeteilt werden. Auch die Umgebungsfehler können wie bei den Anforderungen weiter eingeteilt werden in Rohlings- bzw. Werkstückfehler, Bedienfehler sowie Automatisierungs- und Versorgungsfehler. Abgesehen von den zuletzt genannten Versorgungsfehlern

können diese Gruppen weiter untergliedert werden: Die Rohlings- bzw. Werkstückfehler setzen sich bspw. zusammen aus Geometrie- und Materialfehlern. Diese beiden Fehlerarten können unter anderem auftreten, wenn ein Rohteil in der falschen Größe an eine Werkzeugmaschine übergeben wird oder wenn das Rohteil nicht die erwarteten Materialeigenschaften aufweist. Auch Bedienfehler lassen sich weiter in Interaktions- und Datenfehler zerlegen. Erstgenannte beschreiben Fehler, bei denen ein Maschinenbediener physisch in unzulässiger Weise mit der Werkzeugmaschine interagiert, zum Beispiel durch das Öffnen einer Schutztüre während eines Bearbeitungsvorgangs. Datenfehler hingegen können auftreten, wenn ein Bediener an der graphischen Mensch-Maschine-Schnittstelle falsche bzw. unzulässige Eingaben tätigt (bspw. bei der Einstellung von Maschinenparametern). Auch hinsichtlich der Automatisierungsfehler bietet sich eine weitere Unterteilung in Daten- und Geometriefehler an. Datenfehler beschreiben dabei ein unzulässiges Verhalten in der Kommunikation zwischen Automatisierung und Werkzeugmaschine, bspw. durch einen falschen Startbefehl für einen Bearbeitungsvorgang. Geometriefehler betreffen hingegen die Materialflussschnittstelle, an der bspw. Lageabweichungen bezüglich des zu übergebenden Werkstückes auftreten können. Bei Versorgungsfehlern bietet sich eine weitere Unterteilung allerdings nicht an. Als Beispiele für Fehler in dieser Kategorie können grundsätzlich ein Energieausfall oder ein Defekt der Kühlmittelversorgung genannt werden.

Die inhärenten Fehler einer Werkzeugmaschine lassen sich ähnlich wie bei den Anforderungen den einzelnen Entwicklungsdisziplinen entsprechend in mechanische, elektrische und informationstechnische Fehler einteilen. Die erstgenannten lassen sich anschließend weiter untergliedern in Werkzeugfehler, wie zum Beispiel bei dem Defekt eines Fräskopfs, in Sensorfehler, die durch Nässe oder Verschmutzung hervorgerufen werden können, in Aktorfehler, wie bspw. beim Bruch einer Antriebsachse, sowie in Konstruktionsfehler, die unter anderem zu Schwingungen oder Wärmeentwicklungen im Betrieb führen können. Elektrische Fehler hingegen setzen sich im Wesentlichen aus Übertragungsfehlern zusammen, die bspw. aus Kabelbrüchen oder einer falschen Verdrahtung resultieren können. Informationstechnische Fehler schließlich lassen sich im Wesentlichen durch Funktionsfehler beschreiben, die sich in der Regel aus einer falschen Implementierung ergeben.

2.3 Schlussfolgerungen

Auf Basis der Ergebnisse der Anforderungsanalyse konnten bereits in der frühen Phase des Forschungsprojektes IMoMeSA wichtige Schlussfolgerungen abgeleitet werden, welche die inhaltlichen Arbeiten zum Aufbau einer modellbasierten Entwicklungsmethodik wesentlich beeinflussten. Die zentralen Schlussfolgerungen aus der Anforderungsanalyse werden im Folgenden kurz zusammengefasst:

1.) Ein zentrales Hemmnis in aktuellen Entwicklungsprozessen liegt in der unzureichenden **interdisziplinären Abstimmung** zwischen allen beteiligten Entwicklern. Vor diesem Hintergrund muss es ein modellbasierter Entwicklungsansatz insbesondere ermöglichen, alle Disziplinen durch einen gemeinsamen Modellaufbau möglichst frühzeitig aufeinander abzustimmen. Dabei gilt es insbesondere, die Kompetenzen und Fähigkeiten der einzelnen Disziplinen in einem gemeinsamen Modell der Werkzeugmaschine abzubilden, um so den interdisziplinären Austausch bestmöglich fördern zu können.

2.) Damit sich eine modellbasierte Entwicklungsmethodik langfristig im Werkzeugmaschinenbau etablieren kann, muss eine **nahtlose und ganzheitliche Einbettung** in bestehende Prozesse ermöglicht werden. Aktuell wird von den Unternehmen des projektbegleitenden Ausschusses der Aufwand einer zusätzlichen Modellerstellung und -pflege als zu großes Hemmnis angesehen, welches verhindert, dass derartige Methoden eingesetzt werden könnten. Nur wenn es gelingt, eine Vielzahl an Entwicklungsaufgaben entlang der gesamten Prozesskette durch ein zusätzliches interdisziplinäres Systemmodell zu vereinfachen, kann ein modellbasierter Entwicklungsansatz erfolgreich im Werkzeugmaschinenbau eingeführt werden.
3.) Neben der notwendigen Einbettung in bestehende Entwicklungsprozesse muss durch einen modellbasierten Ansatz ebenfalls die **Kommunikation zwischen digitalen Werkzeugen** optimiert werden. Die Schnittstelle zum Datenaustausch zwischen eingesetzten Tools wurde im Rahmen der Anforderungsanalyse als derzeit großes Defizit offengelegt. Durch ein zentrales Systemmodell, welches den Datenaustausch zentralisiert und verwaltet, kann die Kommunikation zwischen digitalen Werkzeugen erheblich vereinfacht und verbessert werden.
4.) Vor dem Hintergrund immer komplexer werdender Systeme gewinnt eine **frühzeitige, simulative Absicherung** von Werkzeugmaschinen weiter an Bedeutung. Während sich die Simulation bereits innerhalb einzelner Disziplinen als geeignetes Werkzeug zur Qualitätssicherung etabliert hat, mangelt es aktuell noch an einer Absicherung des interdisziplinären Gesamtkonzeptes. Deswegen muss es ein zentrales Ziel eines modellbasierten Entwicklungsansatzes sein, ein erarbeitetes interdisziplinäres Systemmodell einer Werkzeugmaschine auch mittels Simulation zu testen und zu optimieren.
5.) Die Aufgaben des **Anforderungs- und Fehlermanagements** sind mittlerweile essenzielle Bestandteile, um Werkzeugmaschinen trotz steigender Komplexität zielgerichtet entwickeln zu können. Beide Aufgaben werden allerdings im Werkzeugmaschinenbau aktuell nur unzureichend durch digitale Werkzeuge unterstützt. Deswegen bietet es sich an, sowohl das Anforderungs- als auch das Fehlermanagement als integrative Bestandteile eines modellbasierten Entwicklungsprozesses zu betrachten und entsprechende Mechanismen zu etablieren, um diese beiden Aufgaben während des Aufbaus eines interdisziplinären Systemmodells integriert und werkzeuggestützt durchführen zu können.

3 Modellbasierte Entwicklungsmethodik

Im dritten Kapitel des Abschlussberichts wird die modellbasierte Entwicklungsmethodik vorgestellt, die im Rahmen von IMoMeSA erarbeitet wurde. Abbildung 9 zeigt das Vorgehen bei der Entwicklung mechatronischer Systeme nach dem IMoMeSA-Ansatz, welches in Anlehnung an den aktuellen Stand im Werkzeugmaschinenbau (vgl. Abschnitt 2.2.1) in drei Schritte (grau) und vier Artefakte (weiß) eingeteilt ist.

Abbildung 9: Modellbasierter Entwicklungsprozess von der Produktidee bis zur fertigen Maschine

Den Ausgangspunkt einer mechatronischen Entwicklung bildet eine Produktidee, bestehend aus einer grob skizzierten und informellen Beschreibung des zu entwickelnden mechatronischen Systems. Aus der Produktidee wird im ersten Schritt, der Konzeption, ein mechatronisches Konzept abgeleitet. Die Konzeptphase dient prinzipiell der Klärung von Anforderungen zwischen Kunde und Werkzeugmaschinenbauer sowie der Skizzierung von Lösungsideen aller beteiligten Disziplinen. Ein besonderes Augenmerk liegt dabei auf der mechatronischen Modularisierung der Lösungsidee sowie der interdisziplinären Zusammenarbeit zwischen Mechanikern, Elektrikern und Steuerungstechnikern. Sobald ein zufriedenstellendes mechatronisches Konzept gefunden wurde, kann die Entwicklung in den Schritt der Verfeinerung übergehen. Diese hat grundsätzlich das Ziel, das mechatronische Konzept zu einem virtuellen Prototyp der Maschine weiterzuentwickeln. Der IMoMeSA-Ansatz versucht dabei, möglichst viele Teile des mechatronischen Konzepts in der Verfeinerung wiederzuverwenden und ggf. durch genauere Modelle zu erweitern bzw. zu ersetzen. Sobald ein funktionstüchtiger virtueller Prototyp erstellt wurde, kann die Entwicklung in den Schritt des physikalischen Aufbaus übergehen. Dieser Schritt endet, sobald die Bauteile gefertigt bzw. zugeliefert sowie die Maschine montiert und in Betrieb genommen ist. Dabei setzt der IMoMeSA-Ansatz insbesondere auf eine beschleunigte Inbetriebnahme, da viele Probleme schon während der Konzeption und der Verfeinerung erkannt und behoben werden können.

Im Folgenden werden die Schritte der Konzeption (Abschnitt 3.1) und der Verfeinerung (Abschnitt 3.2) genauer beschrieben. Insbesondere wird auf die Modellierungstechnik eingegangen, die dem modellbasierten Entwicklungsprozess zugrunde liegt. Ein weiterer Fokus liegt auf möglichen Qualitätssicherungsmaßnahmen sowie dem systematischen Vorgehen bei der Entwicklung. Der physikalische Aufbau wird schließlich nicht weiter beleuchtet, da diese Entwicklungsphase nicht mehr durch die Einführung einer modellbasierten Entwicklungsmethodik unterstützt wird.

3.1 Konzeption

Wie in Abbildung 10 dargestellt, umfasst der Schritt der Konzeption die typischen Phasen des Systems Engineerings – Analyse, Design und Implementierung [22]. Dabei kommt ein im Projektverlauf entwickelter Editor, der „IMoMeSA Modeller“, zum Einsatz, mit dessen Hilfe die Inhalte des Konzeptes erarbeitet werden können. Dieser ist speziell auf die zugrundeliegende Modellierungstechnik zugeschnitten, welche eine komponentenbasierte Modellierung des mechatronischen Systems vorsieht.

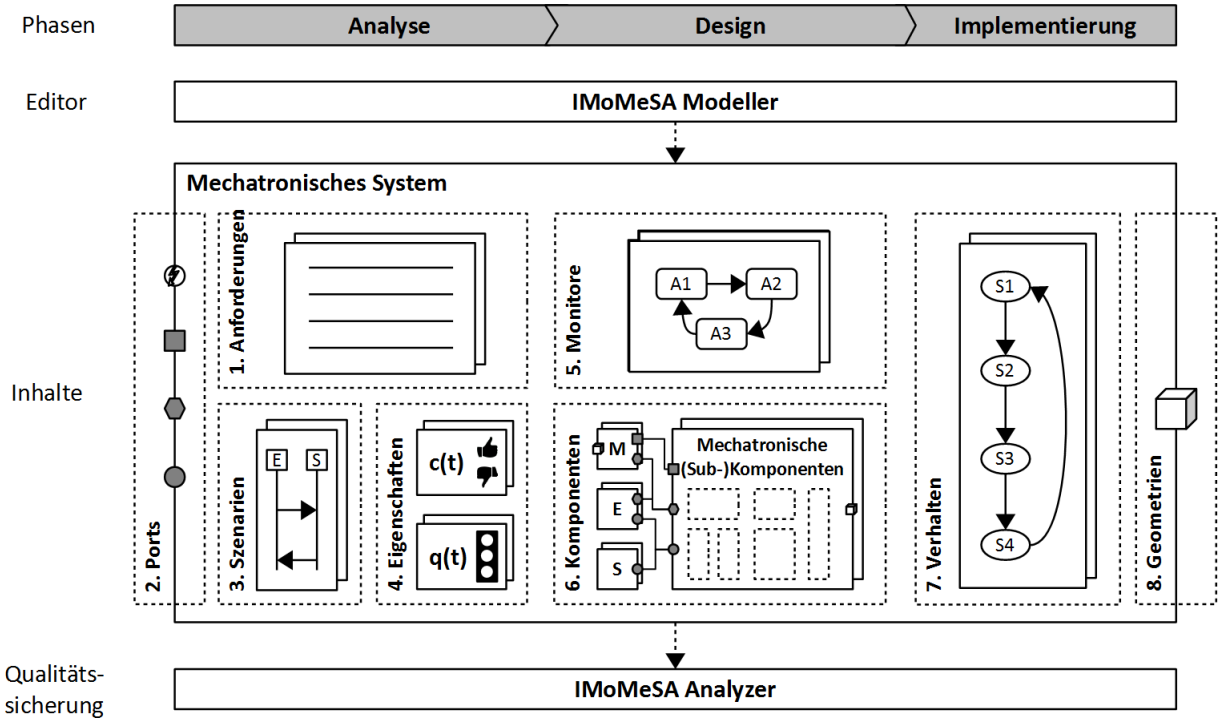


Abbildung 10: Übersicht über den Schritt der Konzeption, bestehend aus Phasen, Editor, den Inhalten sowie der Qualitätssicherung

Die Modellierungstechnik selbst beinhaltet während der Analyse zunächst die Spezifikation von informellen Anforderungen, auf deren Basis die grundsätzliche Schnittstelle des mechatronischen Systems zu seiner Umgebung in Form sog. Ports bestimmt werden kann. Diese können anschließend verwendet werden, um Szenarien zu definieren, welche die geforderte Interaktion des Systems mit seiner Umgebung beschreiben. Schließlich können während der Analyse dem modellierten System noch Eigenschaften hinzugefügt werden, um zuvor modellierte Anforderungen zu formalisieren. Dafür kommen im Wesentlichen Bedingungen und Qualitäten zum Einsatz, die über mathematische Ausdrücke Voraussetzungen beschreiben, denen das zu entwickelnde System genügen muss. Während ein mechatronisches System in der Analyse-Phase primär aus Sicht der Umgebung betrachtet wird, fokussiert die Design-Phase anschließend das System selbst. Dabei können zunächst sog. Monitore spezifiziert werden, die einzelne Zustände definieren, welche das mechatronische System einnehmen kann. Diese können anschließend bei der Entscheidung berücksichtigt werden, ob das gewünschte Systemverhalten zu komplex ist, um es direkt zu implementieren. In diesem Fall kann das System in beliebige (Sub-)Komponenten zerlegt werden. Jede dieser Komponenten kann anschließend nach demselben Entwicklungsvorgehen erstellt werden. Dadurch wird eine hierarchische Struktur des mechatronischen Systems erzeugt, was eine Dekomposition in losgelöste und weniger komplexe Anlagenmodule ermöglicht. Darüber hinaus können die Ports modellierter Komponenten mittels sog. Kanäle verbunden werden. In der Implementierungsphase werden schließlich das Verhalten und die Geometrie einzelner Komponenten modelliert. Verhaltensbeschreibungen beruhen wie Monitore auf diskreten Zustandsautomaten, können allerdings anhand ihrer Verwendung unterschieden werden: Während Monitore lediglich Ein- und Ausgaben überwachen, werden bei der Definition von Verhalten Ausgaben geschrieben, wodurch das tatsächliche

Systemverhalten spezifiziert und simuliert werden kann. Hinsichtlich der Geometrien können dem mechatronischen System bzw. seinen Komponenten noch sog. Bauteile hinzugefügt werden. Diese definieren über einfache Starrkörper und Verbindungen den mechanischen Aufbau des zu entwickelnden Systems.

Zur Qualitätssicherung können modellierte mechatronische Systeme im sog. „IMoMeSA Analyzer“ getestet werden. Darin kommen unterschiedliche Qualitätssicherungsmaßnahmen (z.B. Kompilierung oder Simulation) zum Einsatz, um das mechatronische System bzw. seine Komponenten auf syntaktische und semantische Korrektheit zu überprüfen. Ein besonderer Fokus liegt dabei auf der Konsistenz zwischen gefordertem und implementiertem Verhalten von Komponenten.

Im Folgenden wird zunächst die Modellierungstechnik, mit den eben bereits eingeführten Inhalten, näher betrachtet (Abschnitt 3.1.1). Anschließend werden die Maßnahmen beschrieben, die für die Qualitätssicherung während der Konzept-entwicklung zur Verfügung stehen (Abschnitt 3.1.2). Darauf aufbauend wird ein systematischer Arbeitsablauf erläutert, der die Phasen Analyse, Design und Implementierung sowie den Einsatz der bereitgestellten Techniken beschreibt (Kapitel 3.1.3). Schließlich ist ein abschließender Abschnitt der modellbasierten Konzeption von Fehlerdiagnosemechanismen gewidmet, die zur Erkennung von möglichen Verhaltensfehlern eingesetzt werden können (Abschnitt 3.1.4).

3.1.1 Modellierungstechnik

Die nachfolgende Darstellung der einzelnen Inhalte der Modellierungstechnik konzentriert sich grundsätzlich auf das zugrunde liegende Meta-Modell. Die entsprechenden Ausschnitte dieses Meta-Modells sind dabei als Unified-Modeling-Language-Klassendiagramme mit Generalisierungs- und Aggregationsbeziehungen gezeigt [23].

Des Weiteren sei an dieser Stelle bereits darauf hingewiesen, dass die Modellierungstechnik absichtlich von vielen Details eines realen mechatronischen Systems wie bspw. Schwingungen und Steifigkeiten abstrahiert. Diese Abstraktion wurde gewählt, um in der frühen Phase des mechatronischen Entwicklungsprozesses nicht den Fokus auf das Wesentliche zu verlieren. Als wesentlich wurden bereits im Vorgängerprojekt AutoVIBN [15] der kollisionsfreie Materialfluss und die gezielte Übertragung von Bearbeitungsenergie (z.B. durch Prozesse wie Fräsen oder Schleifen) sowie deren Steuerung über Sensoren und Aktoren identifiziert. Der Materialfluss beinhaltet dabei sowohl gewünschte Werkstück- als auch notwendige Werkzeugbewegungen, die gemeinsam für die Umsetzung des gewünschten Bearbeitungsprozesses sorgen.

Nachfolgend werden nun die einzelnen Inhalte der Modellierungstechnik vorgestellt. Dabei wird grundsätzlich die Reihenfolge aus Abbildung 10 beibehalten, lediglich die Komponente wird an den Beginn der Erläuterungen gestellt, da diese innerhalb des Meta-Modells eine zentrale Rolle einnimmt.

3.1.1.1 Komponenten

Wie bereits zuvor erwähnt, stützt sich der IMoMeSA-Ansatz auf eine komponenten-basierte und hierarchische Modellierung von mechatronischen Systemen. Der Grund für diese Entscheidung ist die damit einhergehende Möglichkeit, Systeme in weniger

komplexe Module (bzw. Komponenten) zerlegen zu können. Außerdem erlaubt die komponentenbasierte Modellierung prinzipiell die Wiederverwendung von bereits entwickelten Teillösungen. Die Modellierungstechnik unterscheidet prinzipiell nicht zwischen mechatronischen und disziplinspezifischen Komponenten. Der Grund dafür ist die Tatsache, dass Komponenten typischerweise zunächst einer zu realisierenden Funktion entsprechen, die erst im Laufe der Entwicklung durch entsprechende Unterkomponenten in ihrem physikalischen Verhalten und somit ihrem konkreten Typ festgelegt werden. Vielmehr ergibt sich der Komponententyp während der Ausgestaltung mit den im Folgenden beschriebenen Modellierungselementen. Man findet jedoch üblicherweise mechatronische Komponenten weiter oben in der Komponentenhierarchie, wohingegen disziplinspezifische Komponenten in der Regel den Unterbau der Komponentenhierarchie stellen. Der Fall, dass eine mechatronische Komponente in einer disziplinspezifischen Komponente enthalten ist, kann somit nicht auftreten. Vielmehr ergibt sich der Typ einer übergeordneten Komponente aus den Typen ihrer Unterkomponenten.

3.1.1.2 Anforderungen

Der erste Schritt bei der Ausgestaltung von Komponenten beinhaltet die (informelle) Modellierung von Anforderungen, welche der Dokumentation von Kundenwünschen dient. Ziel des IMoMeSA-Ansatzes ist es dabei, diese Wünsche möglichst direkt abzubilden, sodass bei der Anforderungsaufnahme bzw. in Kundengesprächen keine unnötigen Aufwände anfallen. Der entsprechende Ausschnitt des Meta-Modells ist in Abbildung 11 gezeigt.

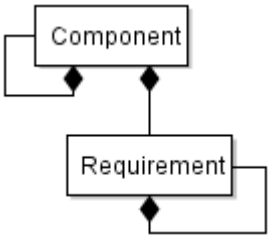


Abbildung 11: Anforderungen im IMoMeSA-Meta-Modell

Prinzipiell können Anforderungen direkt an Komponenten angehängt werden. Anforderungen können somit sowohl für mechatronische als auch für disziplin-spezifische Komponenten festgehalten werden. Dabei werden keine speziellen Arten von Anforderungen unterschieden (wie z.B. Umgebungs- oder Modulanforderungen), sondern alle Arten von Anforderungen über dasselbe Anforderungselement abgebildet. Der Grund dafür ist, dass in Kundengesprächen diese Entscheidung nicht getroffen werden muss. Vielmehr ergibt sich diese Differenzierung durch die aus Anforderungen im Folgenden abgeleiteten Modellierungselemente (z.B. Ports, Eigenschaften oder Szenarien). Um diese Beziehungen festzuhalten, lassen sich Anforderungen mit allen Arten von Modellierungselementen verknüpfen. Auf Meta-Modellebene besitzt eine Anforderung dazu eine Verknüpfung zu der abstrakten Klasse „Element“, von der alle Bestandteile der Modellierungstechnik abgeleitet sind. Aus Übersichtsgründen wurde auf eine Darstellung dieser Klasse verzichtet, weswegen auch die entsprechende Verknüpfung zu den Anforderungen nicht der Abbildung entnommen werden kann.

Durch den Mechanismus der Verknüpfung lassen sich Anforderungen insbesondere auch mit anderen Anforderungen verknüpfen, um zum Beispiel abgeleitete Anforderungen zu dokumentieren. Des Weiteren ermöglicht die Dokumentation von Beziehungen zwischen Anforderungen und anderen Modellierungselementen eine gezielte Aufwandsanalyse im Rahmen des Änderungsmanagements [24].

Für die Erfassung von Anforderungen wurde im Rahmen des Projekts schließlich eine generische Eingabemaske entwickelt. Diese erlaubt die Bearbeitung aller Informationen, die für Anforderungen im Rahmen des IMoMeSA-Ansatzes prinzipiell festgehalten werden können. Welche Informationen dabei in Kundengesprächen und welche in der Nachbereitung erfasst werden, schreibt die Technik nicht vor. Ein grafischer Entwurf der Eingabemaske mit entsprechenden Feldern und Beschriftungen ist in Abbildung 12 dargestellt.

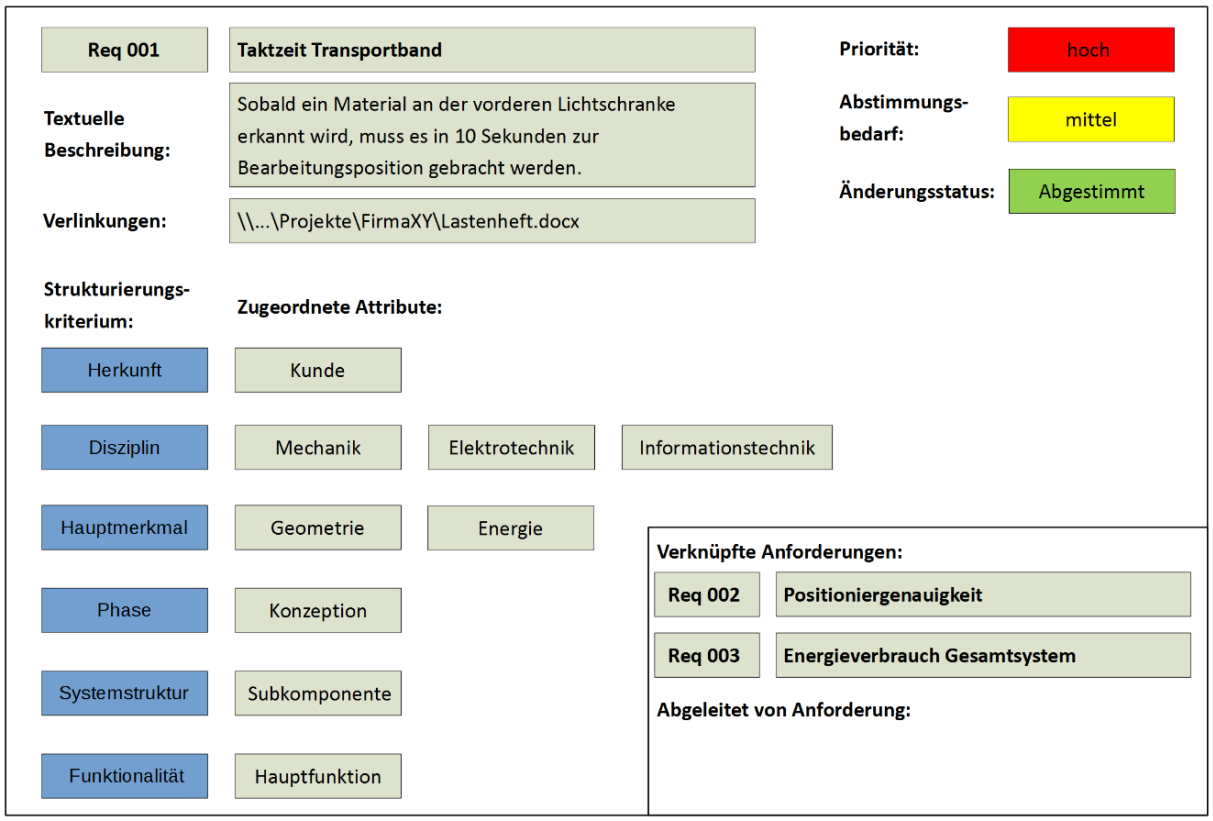


Abbildung 12: Grafischer Entwurf der Eingabemaske für Anforderungen

Neben den zuvor beschriebenen textuellen Elementen und Verknüpfungen enthält die Eingabemaske auch Felder für die Anforderungsstrukturierung und -verwaltung. Für die Strukturierung wurden aus dem Stand der Wissenschaft mögliche Kriterien abgeleitet. Damit ist eine Strukturierung anhand der Herkunft von Anforderungen sowie der betroffenen Disziplinen (z.B. Mechanik), Hauptmerkmale (z.B. Energie), Phasen (z.B. Konzeption), Systemstrukturen (z.B. Subkomponente) und Funktionalitäten (z.B. Hauptfunktion) möglich. Für die Anforderungsverwaltung werden zusätzlich die Felder Priorität, Abstimmungsbedarf und Änderungsstatus bereitgestellt. Während der Änderungsstatus manuell bearbeitet werden muss, können die Priorität und der Abstimmungsbedarf automatisch aus den Strukturierungskriterien abgeleitet werden.

3.1.1.3 Ports

Im Anschluss an die Aufnahme von Anforderungen erfolgt die Modellierung der Schnittstellen der betrachteten Komponente. Der IMoMeSA-Ansatz sieht dafür sog. Ports vor, über welche die Interaktion mit der Umwelt stattfinden kann. Dabei wird im Gegensatz zu bestehenden Techniken explizit zwischen Material-, Energie-, Daten- und Ereignis-Ports unterschieden, um die verschiedenen Disziplinen und Interaktionsmöglichkeiten eines mechatronischen Systems abzudecken. Der entsprechende Ausschnitt des Meta-Modells ist in Abbildung 13 dargestellt.

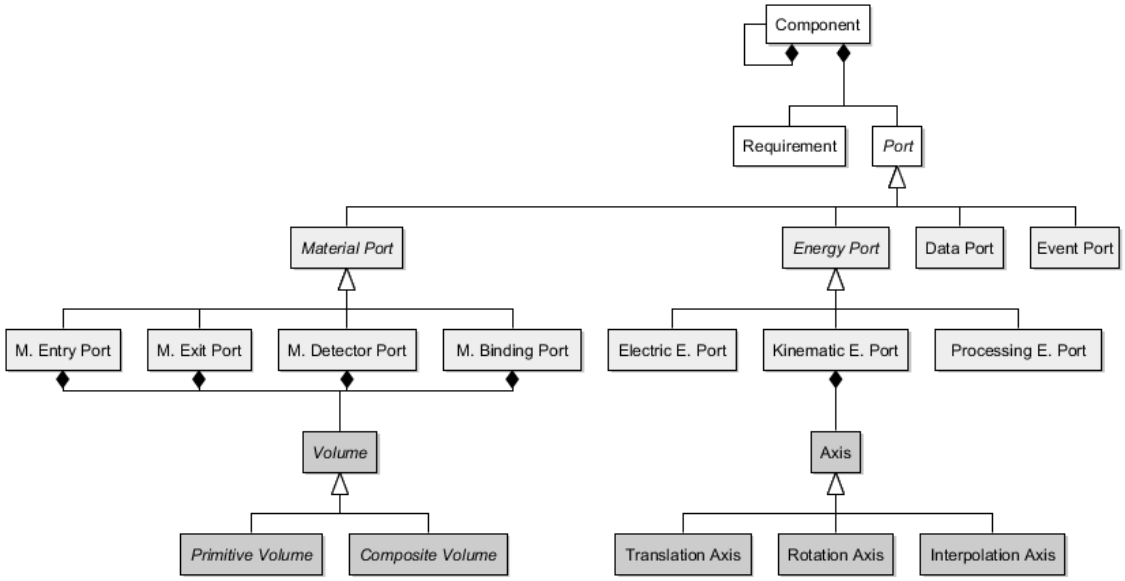


Abbildung 13: Ports im IMoMeSA-Meta-Modell

Wie bereits im Rahmen des Vorgängerprojektes AutoVIBN [15] beschrieben, wird bei Material-Ports des Weiteren zwischen Eingangs- und Ausgangs-Ports sowie Detector- und Binding-Ports unterschieden. Diese Port-Typen haben die Gemeinsamkeit, dass sie durch ein geometrisches Volumen beschrieben sind, welches durch die Ports abgedeckt wird. Der Unterschied liegt hingegen in der Bedeutung der einzelnen Porttypen für das Systemverhalten bzw. die Interaktion mit der Umwelt: Eingangs- und Ausgangs-Ports dienen der dynamischen Erzeugung und Löschung von Komponenten während der Simulation. Damit lassen sich z.B. der An- und Abtransport von Werkstücken sowie die Abspaltung von Spänen während der Werkstück-bearbeitung modellieren. Detector- und Binding-Ports dienen hingegen dem Erkennen von Bauteilen und dem kinematischen Verbinden von Komponenten während der Simulation aufgrund geometrischer Kollisionsbedingungen. Detector-Ports können zum Beispiel für die Modellierung von Lichtschranken verwendet werden, wohingegen Binding-Ports typischerweise für die Modellierung der Interaktion zwischen Transport-bändern und Greifarmen mit Werkstücken eingesetzt werden. Der Unterschied zwischen den beiden Porttypen ist, dass Binding-Ports zusätzlich die Bewegung der bindenden Komponente auf die gebundene Komponente übertragen.

Demgegenüber wird bei Energie-Ports zwischen drei verschiedenen Energietypen unterschieden: Elektrischer Energie, kinematischer Energie und Bearbeitungsenergie. Diese Port-Typen weisen die Gemeinsamkeit auf, dass die Richtung des Energieflusses (d.h. Eingabe oder Ausgabe) während der Konzeptentwicklung festgelegt werden muss. An dieser Stelle wird somit von physikalischen Eigenschaften

wie Masse, Geschwindigkeit und Impuls sowie Spannung, Widerstand und Schaltungstopologie abstrahiert, welche der eigentliche Auslöser für den Energiefluss sind und implizit die Richtung und Größe des Energieflusses festlegen. Diese Abstraktion wurde gewählt, da in der Konzeptphase gewünschte Energieflüsse modelliert werden sollen, während in späteren Phasen deren Umsetzung durch konkrete physikalische Prinzipien im Fokus steht. Im Unterschied zu elektrischen und Bearbeitungsenergie-Ports definieren kinematische Energie-Ports schließlich die konkrete Bewegung über eine entsprechende Transformation. Dabei werden Translations-, Rotations- und Interpolationsachsen unterstützt. Die beiden erstgenannten führen zu einfachen Verschiebungen und Drehungen von Komponenten, während Interpolationsachsen komplexe Bewegungsprofile abbilden können. Details hierzu sind in [15] beschrieben.

Schließlich unterstützt die Modellierungstechnik noch Daten- und Ereignis-Ports. Erstere dienen der Beschreibung von steuerungstechnischen Schnittstellen, während Ereignis-Ports für alle anderen Arten von Umwelteinflüssen und -beeinflussungen verwendet werden können. Sowohl Daten- als auch Ereignis-Ports definieren die Richtung der Interaktion (d.h. Eingabe oder Ausgabe). Des Weiteren spezifizieren sowohl Daten- als auch Ereignis-Ports entsprechende Daten- oder Ereignistypen, welche für entsprechende Kompatibilitätsprüfungen bei Berechnungsvorschriften oder der Verknüpfung von Komponenten verwendet werden können.

3.1.1.4 Eigenschaften

Nachdem die Schnittstellen einer Komponente modelliert sind, können erste Anforderungen formalisiert werden. Über sog. Eigenschaften wird der an den Schnittstellen zu jedem Zeitpunkt beobachtbare Zustand bewertet, wie zum Beispiel der an einem elektrischen Energie-Port anliegende Energiewert. Die Bewertung kann entweder binär (d.h. akzeptabel oder inakzeptabel) oder numerisch (d.h. Grad der Akzeptanz zwischen null und eins) ausfallen. Der dafür konzipierte Ausschnitt des Meta-Modells ist in Abbildung 14 gezeigt.

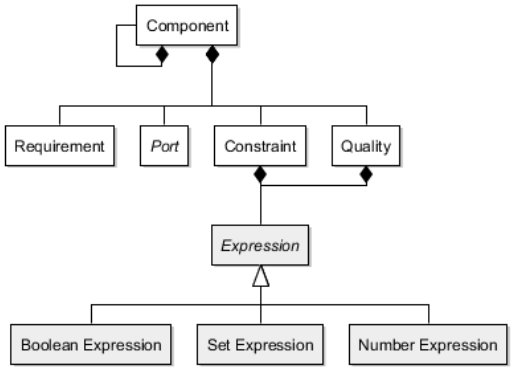


Abbildung 14: Eigenschaften im IMoMeSA-Meta-Modell

Binäre Eigenschaften werden über Bedingungen (engl. Constraints) und Boolesche Ausdrücke modelliert, wohingegen numerische Eigenschaften über Qualitäten (engl. Qualities) und numerische Ausdrücke abgebildet werden. Spezielle Ausdrücke werden dabei für das Lesen der Eingabe- und Ausgabe-Ports bereitgestellt, um somit Portbelegungen in die binären und numerischen Bewertungen einfließen lassen zu können. Während an Energie-Ports in der Regel direkt numerische Werte anliegen, können an Material-Ports Mengen von kollidierenden Komponenten abgegriffen

werden. Aus diesem Grund werden unter anderem auch mengenwertige Ausdrücke wie die Berechnung von Schnittmengen und Mengendifferenzen unterstützt (Set Expressions). Die hier definierten Eigenschaften erlauben somit die Bewertung der Menge der mit einem Material-Port kollidierenden Komponenten genauso, wie die Bewertung von Energie-, Daten- und Ereignisflüssen.

Des Weiteren wurde im Rahmen des Projektes festgestellt, dass Muster bei der Formalisierung von Qualitäten erkennbar sind. Zwei Beispiele dafür sind in Abbildung 15 gezeigt, worin entweder der Maximalwert oder der Zwischenwert einer Beobachtung als optimal bewertet wird. Diese Bewertungen lassen sich allgemein als Funktionen $F(Obs)$ mit den Parametern Min und Max bzw. $Mean$ und Var darstellen. Als Beobachtung (engl. Observation) kann dabei prinzipiell jeder beliebige numerische Ausdruck dienen wie z.B. das Lesen der an einem entsprechenden Energie-Port anliegenden Bearbeitungsenergie. Für diesen Port kann der Fall auftreten, dass eine minimale Energie anliegen muss, um ein gewünschtes Bearbeitungsergebnis zu erzielen (Min). Bis zu einem definierten Maximum an übertragener Energie (Max) steigt die Qualität des Bearbeitungsergebnisses linear an und gilt ab Erreichen dieses Werts als optimal ($F(Obs) = 1$). Andernfalls wäre es auch denkbar, dass ein zu großer Energiewert dazu führt, dass sich das Bearbeitungsergebnis wieder verschlechtert und dass somit nur ein konkreter Wert optimal ist ($Mean$). Ausgehend von diesem Maximum nimmt die Qualität innerhalb einer zulässigen Varianz (Var) linear ab.

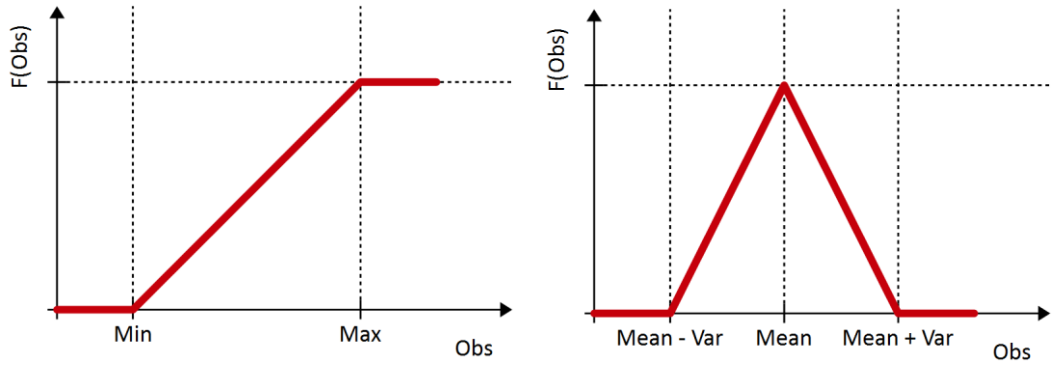


Abbildung 15: Wiederkehrende Muster bei der Formalisierung von numerischen Eigenschaften (Links: Maximalwert, Rechts: Zwischenwert)

Aufgrund der Erkennbarkeit von Mustern bei der Formalisierung von numerischen Eigenschaften liegt es nahe, den IMoMeSA-Ansatz um eine Musterbibliothek für die Modellierung von Eigenschaften zu ergänzen. Diese sollte parametrisierte numerische und Boolesche Funktionen enthalten, welche für die Formalisierung von Eigenschaften verwendet werden können. Einer der Parameter sollte dabei mindestens die Beobachtung sein, über der die Funktion ausgewertet wird. Schließlich sollte noch erwähnt werden, dass ein Zusammenhang zwischen Booleschen und numerischen Mustern besteht. Zum Beispiel deutet sowohl das Muster „Maximalwert ist optimal“ als auch das Muster „Zwischenwert ist optimal“ darauf hin, dass die Beobachtung Obs nie einen bestimmten Schwellwert unterschreiten sollte. Die Unter- und Überschreitung von Schwellenwerten sind ihrerseits Kandidaten für harte Bedingungen (Constraints). Dieses Beispiel zeigt, dass auch eine Vorabverknüpfung zwischen Qualitäten und Bedingungen sinnvoll sein kann, um dem Entwickler bei deren Anwendung Vorschläge für weitere Musteranwendungen machen zu können.

Schließlich sei darauf hingewiesen, dass die hier verwendeten Eigenschaften eine individuelle Bewertung jedes einzelnen Zeitpunkts eines Systemlaufs vornehmen. Bewertungen eines gesamten Systemlaufs können hingegen über entsprechende Aggregationen berechnet werden. Boolesche Eigenschaften können durch logische Konjunktion der Einzelbewertungen aggregiert werden, sodass ein Systemlauf zu jedem Zeitpunkt akzeptiert werden muss, um insgesamt angenommen zu werden. Numerische Bewertungen können demgegenüber statistisch aggregiert werden (z.B. Minimum, Maximum, Durchschnitt, Median, Varianz und Kovarianz). Die relevanten statistischen Indikatoren hängen dabei von der konkreten Eigenschaft ab. Wenn die Eigenschaft zum Beispiel naturgemäß steigt (z.B. größte kumulativ übertragene Bearbeitungsenergie auf das Werkstück während einer Aktivität ist optimal), könnte der erreichte Maximalwert interessant sein. Wenn die Eigenschaft hingegen naturgemäß fällt (z.B. kürzeste Dauer einer Aktivität ist optimal), wäre demgegenüber der erreichte Minimalwert von größerem Interesse.

3.1.1.5 Szenarien

Aufbauend auf den in Abschnitt 3.1.1.4 diskutierten Eigenschaften bieten Szenarien den zweiten wichtigen Baustein des IMoMeSA-Ansatzes zur Formalisierung von Anforderungen. Während Eigenschaften einfache Bedingungen wie die Einschränkung des maximalen Energieverbrauchs zu jedem gegebenen Zeitpunkt erlauben, ermöglichen Szenarien die Formalisierung komplexer Abläufe, die von einem mechatronischen System gefordert werden. Dabei werden die Abläufe aus Sicht der Umgebung des Systems beschrieben, weshalb sie sich auch auf das an der Schnittstelle (bzw. den Ports) beobachtbare Verhalten beziehen. Inspiriert sind Szenarien dabei von Abnahmetests, welche typischerweise zu Projektende im Werkzeugmaschinenbau zum Einsatz kommen, um dem Kunden die korrekte Funktionsweise der Maschine zu demonstrieren. Des Weiteren ist die Modellierung grundsätzlich an Sequenzdiagramme der Unified Modeling Language (UML) [23] sowie an Message Sequence Charts der International Telecommunication Union (ITU) [25] angelehnt. Der entsprechende Ausschnitt des IMoMeSA-Meta-Modells ist in Abbildung 16 gezeigt.

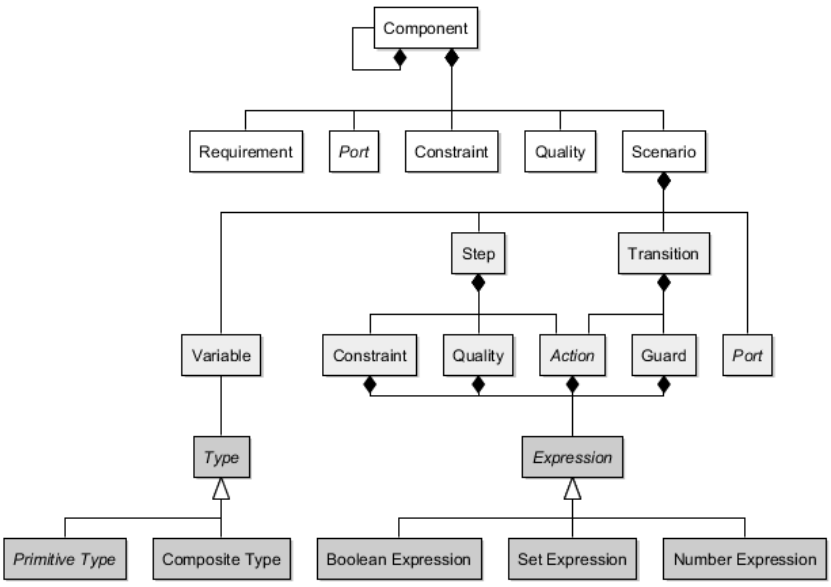


Abbildung 16: Szenarien im IMoMeSA-Meta-Modell

Ein grundlegender Unterschied zwischen UML-Sequenzdiagrammen bzw. Message Sequence Charts und IMoMeSA-Szenarien ist, dass letztere keine explizite Modellierung der interagierenden Komponenten erlauben. Vielmehr sind die interagierenden Komponenten implizit durch die Umgebung (engl. Environment) und die zu testende Komponente festgelegt. Dabei wird ebenfalls implizit angenommen, dass die Schnittstelle der Umgebungskomponente das passende Gegenstück zur Schnittstelle der mechatronischen Systemkomponente bietet. Das heißt, dass die Umgebungskomponente automatisch Ein- bzw. Ausgabe-Ports für entsprechende Aus- bzw. Eingabe-Ports der mechatronischen Systemkomponente bereitstellt. Die Schnittstelle der Umgebungskomponente kann des Weiteren noch um Materialentry- und -exitports erweitert werden, um für das jeweilige Szenario spezifische Komponentenerzeugungen oder -entfernungen abzubilden. Zum Beispiel können über diesen Mechanismus die Positionen für den Ein- und Ausgang von Werkstücken in die bzw. aus der Maschine festgelegt werden, wodurch unter anderem auch die korrekte Funktionsweise der Maschine bei unterschiedlichen Positionierungsgenauigkeiten geprüft werden kann. Schließlich können für Szenarien noch Variablen definiert werden, die über den Ausführungsverlauf für das Zwischenspeichern von Informationen eingesetzt werden können. Ähnlich wie Daten-Ports werden Variablen durch einen entsprechenden Datentyp genauer spezifiziert. Grundsätzlich wird dabei zwischen primitiven und zusammengesetzten Datentypen unterschieden.

Das eigentliche Prüfverhalten wird demgegenüber im Sinne von Schritten und Transitionen modelliert. Schritte können zum Beispiel das Zuführen oder das Bearbeiten von Werkstücken umfassen und decken somit ganze Zeitintervalle eines Prüfablaufs ab. Während des Verbleibs in einem Schritt können Eigenschaften (vgl. Abschnitt 3.1.1.4) ausgewertet und Aktionen durchgeführt werden. Durch Eigenschaften lässt sich zum Beispiel die maximale Dauer eines Schritts einschränken oder die Energieeffizienz während der Durchführung bewerten. Die zugrunde liegenden Ausdrücke können dabei sowohl auf die Port- als auch auf die Variablenbelegungen zugreifen. Aktionen können hingegen verwendet werden, um während der Schrittausführung die Belegungen von Ports oder Variablen zu verändern. Damit kann zum Beispiel die Dauer eines Schritts berechnet oder eine konstante elektrische Energie an das mechatronische System übergeben werden. Schließlich können zwei spezielle Schritte als Start und Ende eines Szenarios ausgezeichnet werden. Um zwischen Schritten zu wechseln, können letztlich Transitionen eingesetzt werden. Transitionen werden durch einen Wächter (engl. Guard) und Aktionen genauer spezifiziert. Der Wächter kann dazu verwendet werden, die Bedingung zu modellieren, die beim Übergang zwischen zwei Schritten gelten muss. Zum Beispiel kann die Bedingung angegeben werden, dass ein Werkstück in einem definierten Zustand der Bearbeitung an einer bestimmten Position zu finden ist. Aktionen können hingegen wie bei Schritten dazu eingesetzt werden, die Belegung von Ports und Variablen zu verändern. Dadurch können zum Beispiel entsprechende Variablen für die Berechnung der Dauer von Schritten zurückgesetzt oder notwendige Datensignale für das Anstoßen des nächsten Bearbeitungsschrittes gesendet werden.

Die Prüfsemantik von Szenarien ist schließlich in Abbildung 17 am Beispiel eines erfüllten und eines nicht erfüllten Szenarios dargestellt. Die gewählte Semantik ist ähnlich der Bedeutung von endlichen Automaten für die Erkennung von Sprachen über

einem definierten Alphabet [26] mit der Besonderheit, dass genau ein akzeptierender und ein nicht akzeptierender Zustand definiert ist. Der akzeptierende Zustand ist dabei identisch mit dem Endschritt des jeweiligen Szenarios, während der nicht akzeptierende Zustand in Szenarien nicht explizit modelliert wird, sondern vielmehr implizit in jedem Szenario enthalten ist. Der akzeptierende Zustand wird während einer Szenario-Ausführung erreicht, wenn eine Folge von Schritten bzw. Transitionen gegangen wurde, die vom Start- zum Endschritt führt und dabei keine Bedingungen der jeweils aktiven Schritte verletzt wurden. Dazu müssen zum Beispiel alle Bedingungen an die maximale Dauer von Schritten oder den maximalen Energieverbrauch der Maschine eingehalten werden. Der nicht akzeptierende Zustand wird hingegen erreicht, wenn in einer entsprechenden Schritt- bzw. Transitionsfolge die Bedingungen mindestens eines aktiven Schritts verletzt wurden. Dies kann zum Beispiel der Fall sein, wenn ein Schritt länger als die maximal vorgeschriebene Dauer aktiv ist oder ein zu hoher Energieverbrauch der Maschine beobachtet wurde. An Schritte geheftete Bedingungen dienen demnach implizit als Übergänge in den nicht akzeptierenden Zustand.

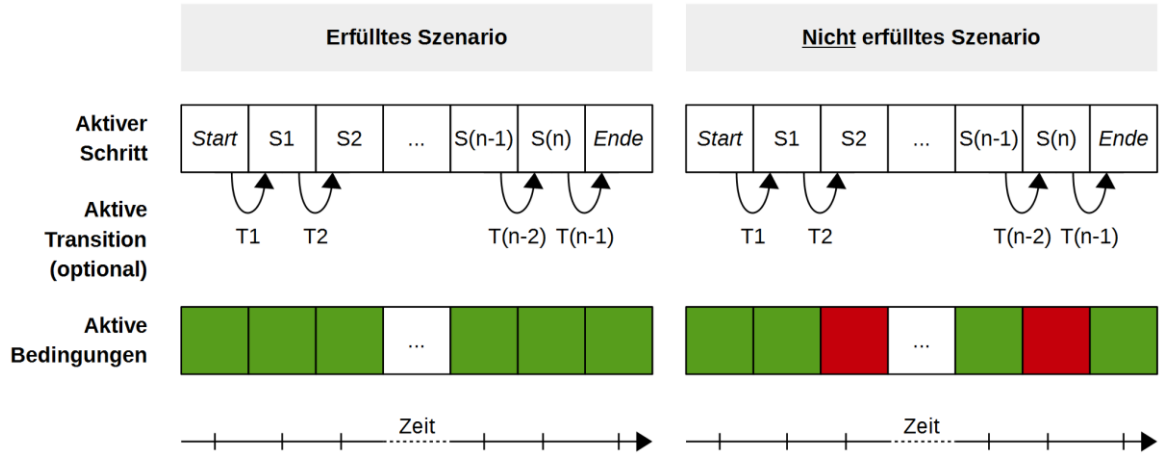


Abbildung 17: Prüfsemantik von Szenarien als Folge von aktiven Schritten

Abschließend ist zu erwähnen, dass die gewählte Modellierung von Szenarien im Grunde eine automatenbasierte Modellierung des Verhaltens der Umgebungskomponente widerspiegelt. Ein ähnlicher Ansatz wird bei der generellen Verhaltensmodellierung vom Komponenten verfolgt (siehe Abschnitt 3.1.1.8), jedoch mit dem Unterschied, dass es weder akzeptierende noch nicht akzeptierende Zustände gibt und somit auch kein Endzustand ausgezeichnet werden muss sowie keine Zustandseigenschaften (d.h. Bedingungen und Qualitäten) definiert werden müssen.

3.1.1.6 Monitore

Die zuvor vorgestellten Modellierungselemente sind grundsätzlich der Analysephase zuzuordnen und widmen sich somit der Betrachtung des Systems aus der Perspektive seiner Umgebung. Der nächste Schritt innerhalb der hier erarbeiteten Entwicklungsmethodik beinhaltet nun die Gestaltung des technischen Prozesses, der auf der Maschine ablaufen soll. Der IMoMeSA-Ansatz sieht dafür eine formale Prozessmodellierung vor, deren Meta-Modell in Abbildung 18 dargestellt ist. Der Prozess wird in Form eines Monitors für das Maschinenverhalten modelliert.

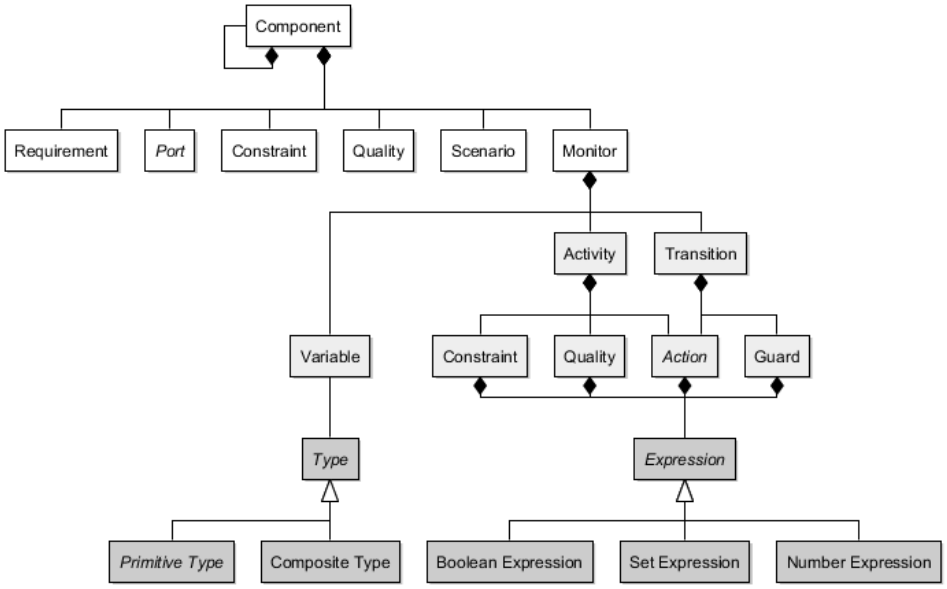


Abbildung 18: Monitore im IMoMeSA-Meta-Modell

Ähnlich zu Szenarien (vgl. Abschnitt 3.1.1.5) können für Monitore typisierte Variablen definiert werden, um Informationen über den Systemverlauf zu verfolgen. Mit Hilfe von Variablen lässt sich zum Beispiel die Dauer oder der aggregierte Energieverbrauch von festgelegten Teilabläufen modellieren. Der eigentlich zu überwachende Prozess wird hingegen ähnlich zu Szenarien mit Aktivitäten (statt Schritten) und Transitionen modelliert. Beispiele für Aktivitäten sind das Fräsen oder Schleifen von Werkstücken sowie das Wechseln von Werkzeugen. Aktivitäten umfassen somit jeweils ein Zeitintervall, während dessen viele kleine Operationen auf der Maschine ausgeführt werden, um diese Aktivität zu realisieren. Des Weiteren können Eigenschaften und Aktionen an Aktivitäten gehängt werden. Mit Hilfe von Eigenschaften lassen sich zum Beispiel Bedingungen wie die maximale Dauer einer Aktivität oder Qualitäten wie die Energieeffizienz einer Aktivität modellieren. Die zugrunde liegenden Booleschen und numerischen Ausdrücke können dafür auf die Belegung von Ein- und Ausgabe-Ports der (mechatronischen) Komponente sowie der Monitorvariablen zugreifen. Mit Hilfe von Aktionen lässt sich hingegen die Belegung der Monitorvariablen während der Ausführung verändern. Transitionen erlauben den Übergang zwischen Aktivitäten und können über Wächter und Aktionen genauer spezifiziert werden. Wächter modellieren die für einen Übergang erforderlichen Bedingungen. Zum Beispiel wäre eine mögliche Übergangsbedingung zwischen den Aktivitäten Fräsen und Schleifen, dass ein gefrästes Werkstück an einer bestimmten Position vorgefunden wird. Aktionen an Transitionen erlauben schließlich, Variablen zurückzusetzen und für die Folgeaktivität zu initialisieren. Zum Beispiel kann die Dauer von Aktivitäten als numerische Variable modelliert werden, die bei jedem Übergang zwischen Aktivitäten auf null gesetzt wird.

Die Semantik von Monitoren ist analog zur Semantik der zuvor eingeführten Szenarien. Grundsätzlich werden Monitore parallel zum Komponentenverhalten ausgeführt. Dabei wird zu jedem Simulationsschritt erst überprüft, ob eine Transition aktiv ist. Dies ist der Fall, wenn der Boolesche Ausdruck des Wächters der Transition den Wert wahr liefert. Sofern eine Transition aktiv ist, werden die Aktionen der Transition ausgeführt und die Aktivität gewechselt. Wenn hingegen keine Transition

aktiv ist, werden die Aktionen der Aktivität ausgeführt. Danach werden die Eigenschaften (d.h. Bedingungen und Qualitäten) der Aktivität ausgewertet. Ein Systemlauf erfüllt dabei den spezifizierten Prozess, wenn die Folge von Aktivitäten und Transitionen keine Aktivitätsbedingungen verletzt. Das heißt, dass unter anderem die maximale Dauer aller Aktivitäten und eventuelle Einschränkungen des Energieverbrauchs eingehalten werden müssen. Ein Systemverlauf erfüllt den spezifizierten Prozess hingegen nicht, wenn mindestens eine Aktivitätsbedingung verletzt ist. Das kann zum Beispiel passieren, wenn eine Aktivität die maximal erlaubte Dauer überschreitet oder der kumulative Energieverbrauch zu hoch ist.

Im Grunde stellt die gewählte Modellierung den Prozess als endlichen Automaten [26] mit einem zu akzeptierenden und nur einem impliziten nicht zu akzeptierenden Zustand dar. Dabei wird der nicht zu akzeptierende Zustand erreicht, sobald eine Aktivitätsbedingung verletzt ist. Eine Implementierung des spezifizierten Prozesses wird hingegen akzeptiert, wenn sie in keinem Fall zu Verletzungen von Aktivitätsbedingungen führen kann. Wie in [27] beschrieben, erlauben Verfahren wie Model Checking [28] den vollständigen Beweis, dass eine Implementierung bezüglich einer Prozessspezifikation akzeptiert werden kann. Die Anwendbarkeit dieser Verfahren hängt jedoch von der Komplexität des Zustandsraums des zugrunde liegenden Systemmodells ab. Falls ein vollständiger Beweis der Korrektheit mittels Model Checking nicht möglich ist, kann die Korrektheit zumindest bezüglich modellierter Szenarien mittels Simulation sichergestellt werden.

3.1.1.7 Kanäle

Nach der Spezifikation des technischen Prozesses muss die Entscheidung getroffen werden, ob eine Zerlegung der Komponente notwendig ist oder ob diese direkt implementiert werden kann. Diese Entscheidung orientiert sich grundsätzlich an der Komplexität des gewünschten Verhaltens bzw. der gewünschten Bauteilgeometrie. Je komplexer die gerade betrachtete Komponente dabei ist, desto sinnvoller ist auch eine Zerlegung in entsprechende Module. Als Entscheidungshilfe kann die Regel dienen, dass eine Zerlegung in der Regel dann zielführend ist, wenn für die betrachtete Komponente keine direkte Implementierung über disziplinspezifische Standardkomponenten (z.B. Sensoren/Aktoren) gefunden werden kann. Um Zerlegungen in der Modellierungstechnik abzubilden, wurde eine hierarchische Modellierung der Komponentenbeziehungen gewählt (vgl. Abschnitt 3.1.1.1). Des Weiteren werden für die Interaktion zwischen Komponenten entsprechende Kanäle zur Verfügung gestellt. Der zugehörige Ausschnitt des IMoMeSA-Meta-Modells ist in Abbildung 19 gezeigt.

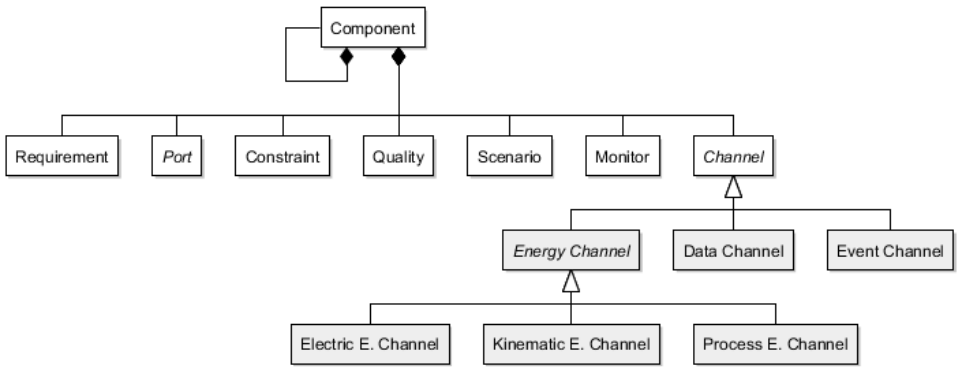


Abbildung 19: Kanäle im IMoMeSA-Meta-Modell

Grundsätzlich werden drei Arten von Kanälen unterschieden: Energiekanäle, Datenkanäle und Ereigniskanäle. Bei ersteren wird des Weiteren zwischen der konkreten Energieform unterschieden, d.h. es können elektrische Energiekanäle, kinematische Energiekanäle und Bearbeitungsenergiekanäle modelliert werden. Alle Arten von Kanälen dienen grundsätzlich der Verbindung von Ports, um den Energie-, Daten- und Ereignisfluss zu definieren. Die verbundenen Ports müssen dazu kompatibel sein, d.h. elektrische Energiekanäle verbinden elektrische Energie-Ports, usw. Eine Ausnahme bilden die kinematischen Energiekanäle, welche einen kinematischen Energie-Port entweder direkt mit einer Komponente oder mit einem Detector- bzw. Binding-Port verbinden. Dabei wird die kinematische Energie entweder direkt auf eine benachbarte Komponente übertragen oder auf die mit einem Material-Port kollidierenden Komponenten. Eine detailliertere Beschreibung der Zusammenhänge kinematischer Energie-Ports, Material-Ports und Komponenten kann in [15] gefunden werden.

3.1.1.8 Verhalten

Für alle Komponenten, bei denen eine weitere Zerlegung in Subkomponenten nicht mehr notwendig ist, kann in der Implementierungsphase schließlich ein konkretes Verhalten spezifiziert werden. Der Begriff Verhalten bezieht sich dabei auf Energie-, Daten- und Ereignisausgaben sowie Materialerzeugung und -entfernung auf Basis entsprechender Material-, Energie-, Daten- und Ereigniseingaben [15]. Des Weiteren sind sowohl das Verhalten im Normalfall als auch mögliche Abweichungen vom Normalverhalten im Fehlerfall eingeschlossen. Wie zuvor dargestellt, stützt sich der IMoMeSA-Ansatz dabei auf eine automatenbasierte Modellierung des Verhaltens. Der entsprechende Ausschnitt des Meta-Modells ist in Abbildung 20 gezeigt.

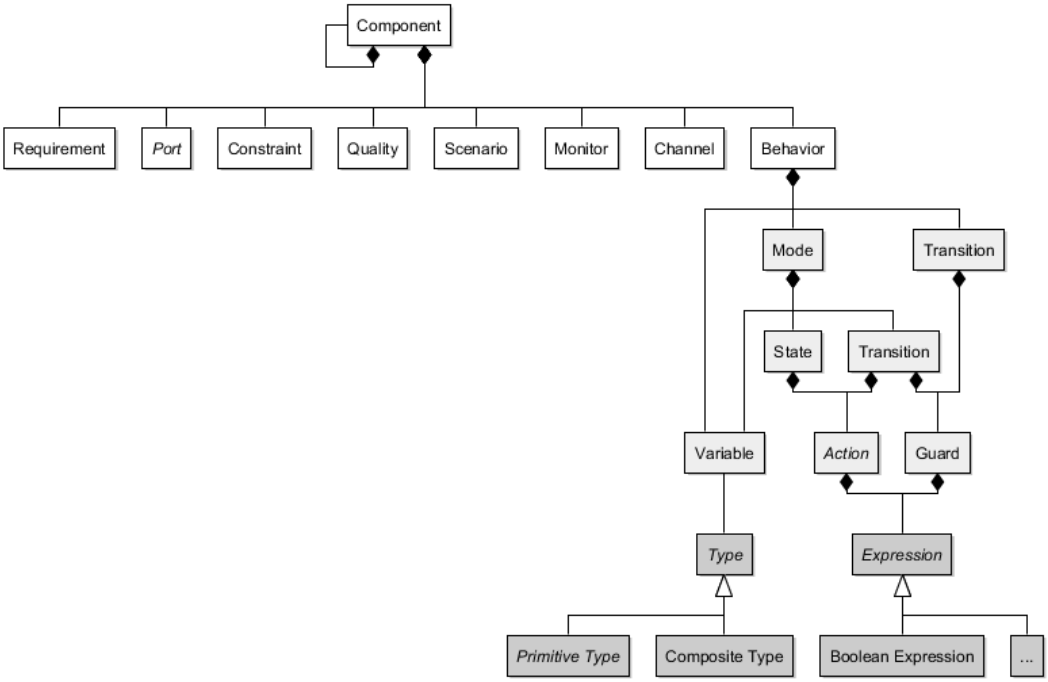


Abbildung 20: Verhalten im IMoMeSA-Meta-Modell

Analog zu Szenarien und Monitoren können für die Spezifikation des Komponentenverhaltens zunächst Variablen mit entsprechenden Datentypen definiert werden. Das Verhalten wird anschließend über Modi und Transitionen spezifiziert. Modi definieren zunächst mögliche Verhaltensausprägungen. Darunter fallen in der Regel das

Verhalten im Normalfall sowie das Verhalten in definierten Fehlerfällen. Des Weiteren sind Übergänge zwischen Verhaltensmodi möglich, welche durch entsprechende Wächter genauer spezifiziert werden. Die einem Wächter zugrunde liegenden Booleschen Ausdrücke können dabei auf die Belegung der Eingabe-Ports sowie der Variablen zugreifen. Typischerweise wird der Übergang zwischen Modi jedoch durch entsprechende Ereignis-Ports gesteuert. Das Verhalten in einem Modus kann schließlich über einen untergeordneten Zustandsautomaten beschrieben werden. Diese Zustandsautomaten können wiederum Variablen definieren, um für den Modus spezifische Informationen zu verfolgen. Das Modus-Verhalten wird hingegen mittels Zuständen und Transitionen modelliert. Zustände können zum Beispiel das Zuführen bzw. Abführen von Material in einer Steuerungskomponente sein. Des Weiteren können sie durch Aktionen genauer spezifiziert werden. Wie zuvor erlauben Aktionen das Verändern von Ausgabe-Port- und Variablenbelegungen. Transitionen erlauben schließlich den Wechsel zwischen Zuständen. Dabei können wieder sogenannte Wächter und Aktionen spezifiziert werden.

Die Semantik der geschachtelten Verhaltensmodellierung ist in Abbildung 21 dargestellt. Grundsätzlich sind für jeden Simulationsschritt ein aktiver Modus und ein aktiver Zustand des untergeordneten Zustandsautomaten definiert. Bei Übergang zum nächsten Simulationsschritt wird zunächst geprüft, ob eine Modus-Transition möglich ist. Dies ist der Fall, wenn der Ausdruck des entsprechenden Transitionswächters den Wert wahr zurück liefert. Ein Beispiel hierfür wäre ein von der Umgebung ausgelöstes Ereignis. Sofern der Modus gewechselt wird, wird der aktive Zustand auf den Startzustand des untergeordneten Zustandsautomaten des neuen Modus gesetzt. Danach muss zunächst geprüft werden, ob eine Zustandstransition möglich ist. Eine Zustandstransition ist möglich, wenn der Ausdruck des entsprechenden Transitions-wächters den Wert wahr zurückliefert. Dies kann zum Beispiel der Fall sein, wenn ein entsprechender Sensorwert gemessen wurde oder eine benachbarte Steuerungs-komponente eine Nachricht gesendet hat. Sofern der Zustand gewechselt wird, werden die Aktionen der Transition ausgeführt, andernfalls werden die Aktionen des Zustands ausgewertet. Aktionen erlauben dabei das Verändern von Ausgabeport- und Variablenbelegungen.

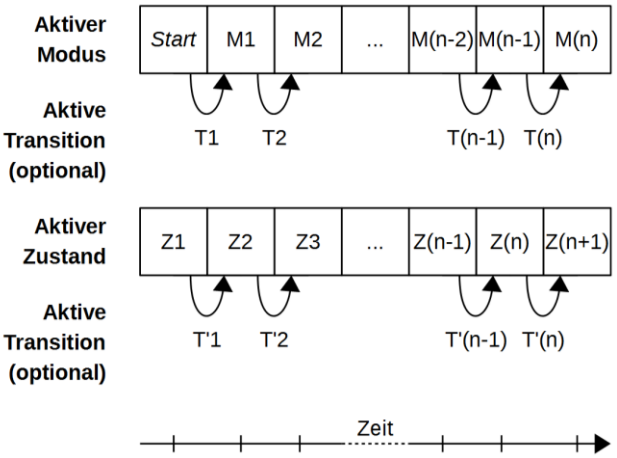


Abbildung 21: Semantik der geschachtelten Verhaltensmodellierung als Folge von aktiven Modi und aktiven Zuständen

Schließlich sei an dieser Stelle angemerkt, dass sich die Modellierung des Komponentenverhaltens kaum von der Modellierung von Szenarien und Monitoren unterscheidet (weder in Syntax, noch in Semantik). Vielmehr wurde darauf geachtet, dass die einzelnen Modellbestandteile einen hohen Deckungsgrad aufweisen. Ziel dabei war es, eine möglichst schlanke Modellierungstechnik zu entwerfen, um die Erlernbarkeit für Entwickler aus unterschiedlichen Disziplinen und somit mit stark unterschiedlichen Vorkenntnissen deutlich zu vereinfachen.

3.1.1.9 Bauteile

Der letzte Bestandteil der Modellierungstechnik beinhaltet die Modellierung der geometrischen Objekte des Systems. Dabei geht der IMoMeSA-Ansatz analog zu anderen Ansätzen wie dem Siemens Mechatronics Concept Designer [29] davon aus, dass in der Konzeptphase hauptsächlich die Modellierung grober Geometrien von Interesse ist, um den grundsätzlich notwendigen Bauraum mechanischer Komponenten zu skizzieren sowie kollisionsfreie Fahrwege sicherzustellen. Nicht betrachtet werden an dieser Stelle hingegen Massen und Reibungskräfte sowie weiterführende physikalische Effekte und Parameter wie Schwingungen und Steifigkeiten. Aus diesem Grund bietet der IMoMeSA-Ansatz einen reduzierten Funktionsumfang für die Bauteil- bzw. Geometriemodellierung im Vergleich zu anderen mechanischen Konstruktionswerkzeugen. Der entsprechende Ausschnitt des Meta-Modells für die Modellierung von Bauteilen ist schließlich in Abbildung 22 gezeigt.

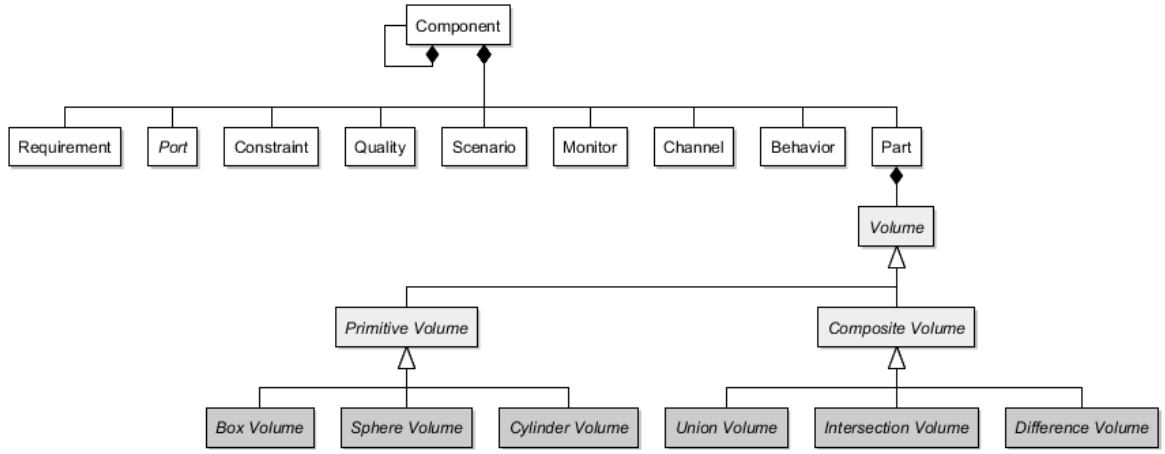


Abbildung 22: Bauteile im IMoMeSA-Meta-Modell

Bauteile können prinzipiell an Komponenten gehängt werden und sind Teil der Schnittstelle (vgl. Abbildung 10). Des Weiteren werden Bauteile analog zu Material-Ports durch die zugrunde liegende Bauteilgeometrie bzw. das entsprechende Volumen genauer spezifiziert. Bei der Geometrie- bzw. Volumenmodellierung wird grundsätzlich der Constructive-Solid-Geometry-Ansatz (CSG) [30] verfolgt (siehe Abbildung 23), welcher die Geometriemodellierung auf Basis von primitiven Volumen und Booleschen Operationen auf Volumen unterstützt. Primitive Volumen sind zum Beispiel Boxen, Kugeln und Zylinder, die in der Regel über wenige Parameter wie die Seitenlänge oder den Durchmesser beschrieben werden können. Boolesche Operationen können hingegen die Vereinigung, Schnittmenge und Differenz anderer Volumen sein. In Abbildung 23 ist exemplarisch die Differenz zweier primitiver Volumen gezeigt.

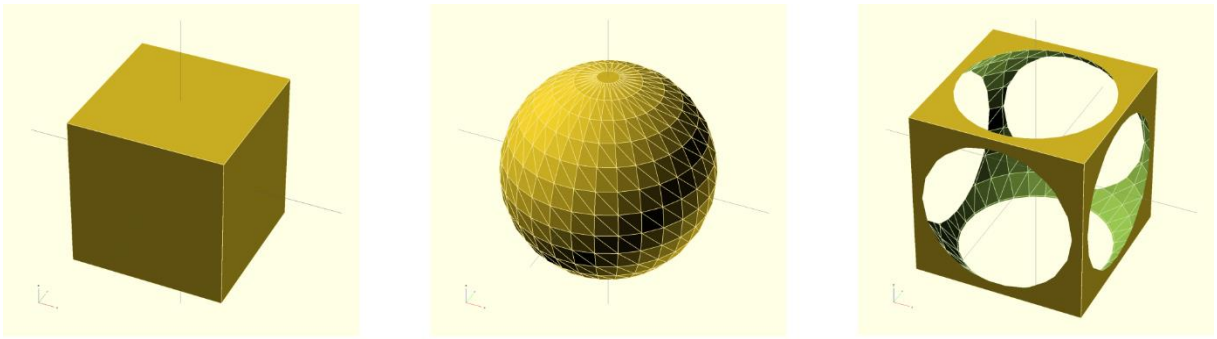


Abbildung 23: Geometriemodellierung nach dem Constructive-Solid-Geometry-Ansatz (CSG) [30]

Abschließend ist zu erwähnen, dass Boolesche Operationen ebenfalls Volumen erzeugen und somit eine komplexe Schachtelung von primitiven Volumen und Booleschen Operationen möglich ist. Außerdem können beliebige Transformationsketten auf Volumen angewendet werden, um zum Beispiel eine Verschiebung oder Drehung im lokalen Koordinatensystem zu erreichen. Dabei definieren die Komponente selbst und alle Booleschen Operationen die Menge der (geschachtelten) lokalen Koordinatensysteme.

3.1.2 Qualitätssicherung

Neben dem Aufbau mechatronischer Systeme mittels der eben eingeführten Modellierungstechnik ist die Qualitätssicherung ein weiterer zentraler Baustein der gesamten modellbasierten Entwicklungsmethodik, die im Rahmen des Forschungsprojektes IMoMeSA erarbeitet wurde. Innerhalb der Qualitätssicherung wurden einzelne Maßnahmen definiert und ausgearbeitet, die dazu beitragen sollen, dass ein entwickeltes Modell des mechatronischen Systems in sich stimmig und korrekt ist. Die dafür bereitgestellten Mechanismen sind in Abbildung 24 dargestellt.

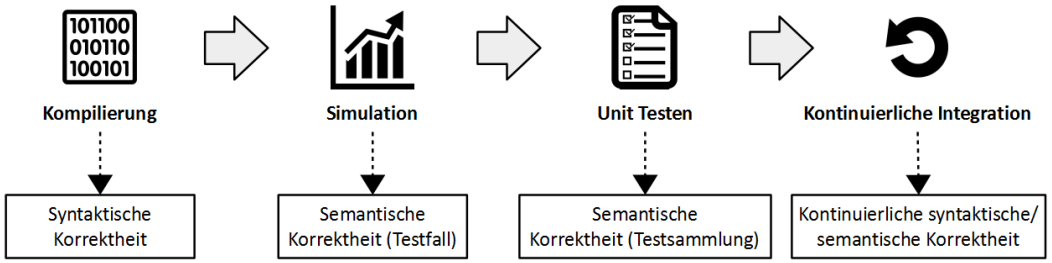


Abbildung 24: Mechanismen innerhalb der IMoMeSA-Qualitätssicherung

Der IMoMeSA-Ansatz bietet eine Reihe von auf einander aufbauenden Mechanismen für die Qualitätssicherung. Der erste Mechanismus ist die Kompilierung, welche auf Basis des statischen Systemmodells und der Modellierungsregeln die syntaktische Korrektheit prüft (vgl. Abschnitt 3.1.2.1). Der zweite Mechanismus ist die Simulation, in der auf Basis eines syntaktisch korrekten Modells ein konkretes Szenario abläuft, wodurch die semantische Korrektheit des Modells bezüglich dieses Testfalls bestimmt werden kann (vgl. Abschnitt 3.1.2.2). Der dritte Mechanismus ist das Unit-Testen [31], welches auf Basis eines syntaktisch korrekten Modells und der Simulation die semantische Korrektheit bezüglich aller Testfälle auswertet (vgl. Abschnitt 3.1.2.3). Der vierte Mechanismus ist schließlich die kontinuierliche Integration [32], welche die

drei zuvor beschriebenen Mechanismen ständig und automatisiert (z.B. stündlich oder täglich) auf einem zentralen Server anwendet, um mögliche Unstimmigkeiten so früh wie möglich zu erkennen und an die Entwickler oder das Management zu melden (vgl. Abschnitt 3.1.2.4). Im Folgenden werden diese Mechanismen detailliert beschrieben.

3.1.2.1 Kompilierung

Die Kompilierung prüft ein statisches (d.h. nicht simuliertes) Systemmodell auf syntaktische Korrektheit bzw. die Einhaltung der Modellierungsregeln. Als Beispiele können an dieser Stelle die Prüfungen genannt werden, ob bei Verbindung zweier Daten-Ports mit einem Datenkanal die entsprechenden Typen kompatibel sind oder ob eine Bedingung durch einen Booleschen Ausdruck berechnet wird (vgl. Abschnitt 3.1.1.4). Die Menge der Modellierungsregeln ergeben sich somit aus den einzelnen Elementen der Modellierungstechnik und deren möglichen bzw. zulässigen Beziehungen, welche bereits zuvor informell beschrieben wurden. Zwei beispielhafte Ergebnisse der syntaktischen Prüfung sind in Abbildung 25 dargestellt.

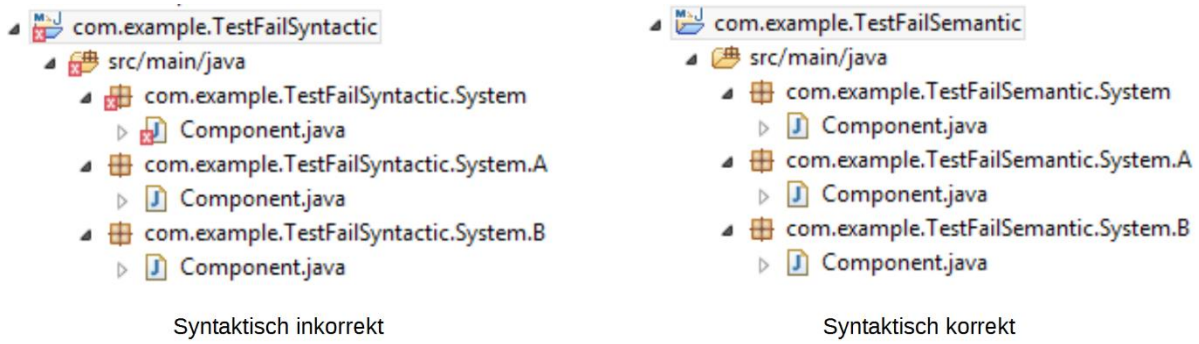


Abbildung 25: Beispielhafte Ergebnisse der Kompilierung

Beide Beispiele setzen sich aus einem System sowie zwei modellierten Unterkomponenten (A und B) zusammen. Im linken Beispiel wurden die Ports dieser Komponenten falsch miteinander verbunden, so dass ein syntaktischer Fehler vorliegt. Dieser wird durch einen entsprechenden Fehlermarker auf der Ebene des Gesamtsystems visualisiert. Allgemein gilt, dass genau das Element mit einem Fehlermarker ausgezeichnet wird, in dem ein syntaktischer Fehler festgestellt wird. Diese Marker werden des Weiteren entlang der Elementhierarchie nach oben propagiert, um eine gezielte Fehlerlokalisation zu ermöglichen.

3.1.2.2 Simulation

Der zweite Mechanismus zur Qualitätssicherung ist die Simulation, welche die semantische Korrektheit des Systemmodells bezüglich eines ausgewählten Szenarios prüft. Vorbedingung der Simulation ist, dass das Systemmodell syntaktisch korrekt ist und somit alle Regeln der Modellierungstechnik eingehalten sind. Sofern dies der Fall ist, kann jedes beliebige Szenario für die Simulation ausgewählt und gestartet werden, um die semantische Korrektheit des Systemmodells unter den modellierten Umständen zu prüfen. Semantische Korrektheit meint dabei, dass während der Simulation keine der Bedingungen (vgl. Abschnitt 3.1.1.4) verletzt werden bzw. die entsprechenden Booleschen Ausdrücke zu keinem Simulationsschritt den Wert falsch liefern. Dies schließt insbesondere auch die Bedingungen ein, welche für das gewählte Szenario und die Monitore definiert wurden. Das beispielhafte Ergebnis eines Simulationslaufs ist schließlich in Abbildung 26 gezeigt.

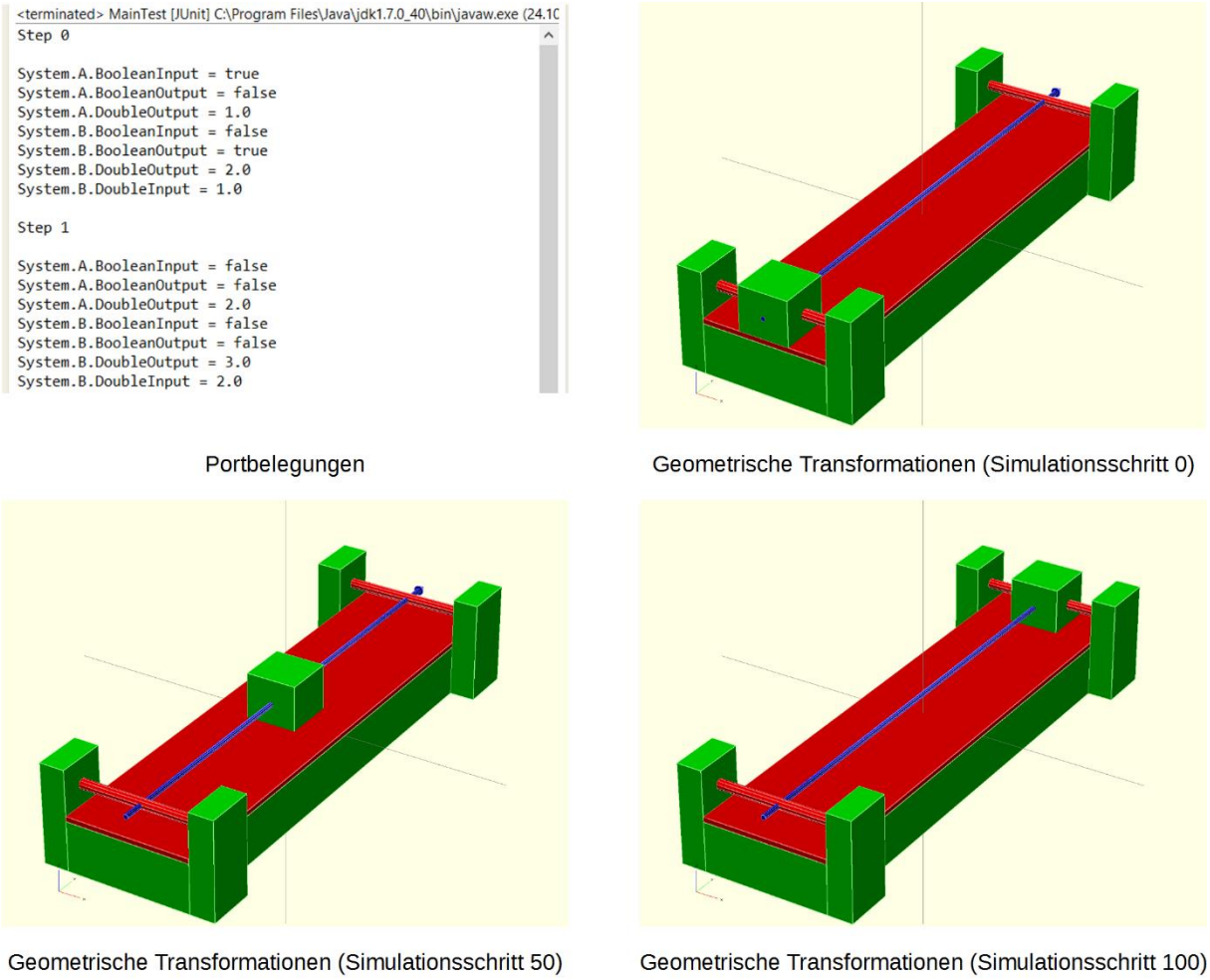


Abbildung 26: Beispielhafte Ergebnisse der Simulation eines syntaktisch korrekten Systemmodells und ausgewählten Szenarios

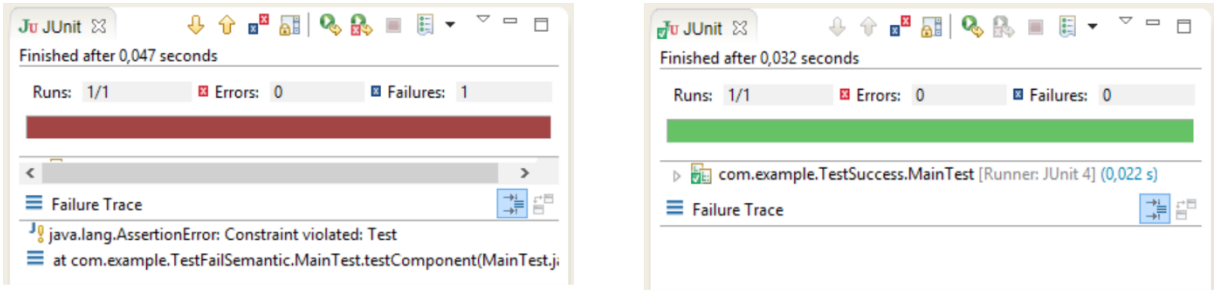
Kern der Simulation ist der Zustand des Systemmodells über die Simulationszeit hinweg. Der Zustand des Systemmodells für einen gegebenen Simulationsschritt ist wiederum durch die Port- und Variablenbelegungen, die geometrischen Transformationen der Komponenten, die Werte der Bedingungen und Qualitäten sowie die aktiven Zustände und Modi, Aktivitäten und Schritte definiert. Abbildung 26 zeigt beispielhaft die Portbelegungen sowie die geometrischen Transformationen der Komponenten über einen Simulationslauf hinweg. In der 3D-Visualisierung werden Material-Ports rot, Achsen blau und Bauteile grün angezeigt, wobei das Systemmodell ein Werkstück, ein Transportband und zwei Positionssensoren enthält. Die Aufgabe des Systems ist das autonome Erkennen von Werkstücken an der Startposition sowie der Transport von erkannten Werkstücken an die Endposition innerhalb einer vordefinierten Zeit und eines vordefinierten Energieverbrauchs. In der Simulation kann überprüft werden, ob das Werkstück korrekt erkannt und an die richtige Stelle transportiert wird und ob der Transport dabei schnell und energieeffizient genug durchgeführt werden kann.

3.1.2.3 Unit-Testen

Aufbauend auf der Kompilierung und Simulation stellt das Unit-Testen den dritten Mechanismus zur Qualitätssicherung im Rahmen des IMoMeSA-Ansatzes dar. Wie bei der Simulation ist die Voraussetzung für die Anwendung dieses Mechanismus,

dass ein syntaktisch korrektes Systemmodell vorliegt und somit alle Modellierungsregeln eingehalten sind. Analog zur Simulation ist es die Aufgabe des Unit-Testens, die semantische Korrektheit des Systemmodells zu prüfen. Im Gegensatz zur Simulation prüft Unit-Testen jedoch die semantische Korrektheit bezüglich aller modellierten Szenarien, anstatt jeweils nur ein ausgewähltes Szenario zu betrachten. Dazu werden alle Szenarien in eine entsprechende Testumgebung geladen und dort automatisiert ausgeführt. Semantische Korrektheit bezüglich aller Szenarien bedeutet schließlich, dass semantische Korrektheit für jedes einzelne Szenario des Systemmodells durch Simulation nachgewiesen werden kann. Ein mögliches Ergebnis des Unit-Testens ist in Abbildung 27 dargestellt.

Hierin ist die prototypische Umsetzung des Unit-Testens mit Hilfe einer Modelltransformation in die Sprache Java [33] und die Entwicklungsumgebung Eclipse [34] gezeigt. Des Weiteren wird das Unit Test Framework JUnit [35] verwendet, um die Testfälle darzustellen und die Testberichte zu generieren. Die Testfälle werden dabei direkt aus den modellierten Szenarien abgeleitet. Ein Testfall schlägt entsprechend fehl, wenn während der Simulation des Szenarios mindestens eine Bedingungsverletzung beobachtet werden kann. Als Rückgabewert liefert der Testfall diejenigen Zeitpunkte und Bedingungen, für die Bedingungsverletzungen beobachtet wurden. Falls hingegen keine Bedingungen verletzt wurden, gilt der Testfall als bestanden. Ein Testbericht fasst die fehlgeschlagenen und die erfolgreichen durchgeführten Testfälle zusammen. Mitunter erlaubt der Bericht auch den Zugriff auf das Ergebnis der individuellen Testfälle und somit auch den Zugriff auf die beobachteten Bedingungsverletzungen. Diese Informationen sollen bei der Behebung von semantischen Fehlern helfen. Sofern schließlich alle Testfälle erfolgreich ausgeführt werden können, gilt die Komponente als erfolgreich implementiert bzw. semantisch korrekt.



Semantisch inkorrekt

Semantisch korrekt

Abbildung 27: Beispielhafte Ergebnisse des Unit-Testens bzw. der semantischen Prüfung bezüglich aller Szenarien

3.1.2.4 Kontinuierliche Integration

Der letzte Baustein der Qualitätssicherung ist die kontinuierliche Integration des mechatronischen Systemmodells, welche die ständige Überwachung der syntaktischen und semantischen Korrektheit forciert. Die kontinuierliche Integration ist besonders wertvoll, wenn mehrere Entwickler parallel am Modell eines mechatronischen Systems arbeiten und somit ungewollt syntaktische und semantische Inkonsistenzen entstehen. Zum Beispiel ist es denkbar, dass ein Elektroingenieur während der Entwicklung einen modellierten Sensor mit seinen zugehörigen Ports austauscht, während ein Steuerungsentwickler gerade eine Ablaufsteuerung für den

Sensor des ursprünglichen Typs erstellt. Beide Entwickler haben dabei in der Regel keinen Zugriff auf die Änderungen des Anderen und können somit in ihren Versionen des Systemmodells keine Inkonsistenzen erkennen. Sobald die Entwickler jedoch ihre Änderungen auf eine zentrale Instanz des Systemmodells überspielen, entsteht entweder direkt ein syntaktischer oder indirekt ein semantischer Fehler. Das Ziel der kontinuierlichen Integration ist es, diese Quellen von Inkonsistenzen bzw. Fehlern so früh wie möglich zu erkennen und zu melden. Die kontinuierliche Integration wendet dazu die bereits beschriebenen Mechanismen Kompilierung und Simulation bzw. Unit-Testen regelmäßig auf eine zentrale Instanz des mechatronischen Systemmodells an. Die Ergebnisse der Mechanismen werden danach in einer Datenbank für den künftigen Zugriff gespeichert. Eine solche Datenbank ermöglicht die Visualisierung von Trends in der Entwicklung, die insbesondere für das Projektmanagement eine hohe Bedeutung besitzen können. Zudem werden nach Anwendung der Mechanismen syntaktische und semantische Fehler direkt per E-Mail an die Entwickler gemeldet. Die Meldungen sollen das schnelle Eingreifen der Entwickler beim Auftreten von Fehlern ermöglichen. Ein Entwurf der Benutzerschnittstelle für den Zugriff auf die Datenbank der kontinuierlichen Integration ist schließlich in Abbildung 28 dargestellt.

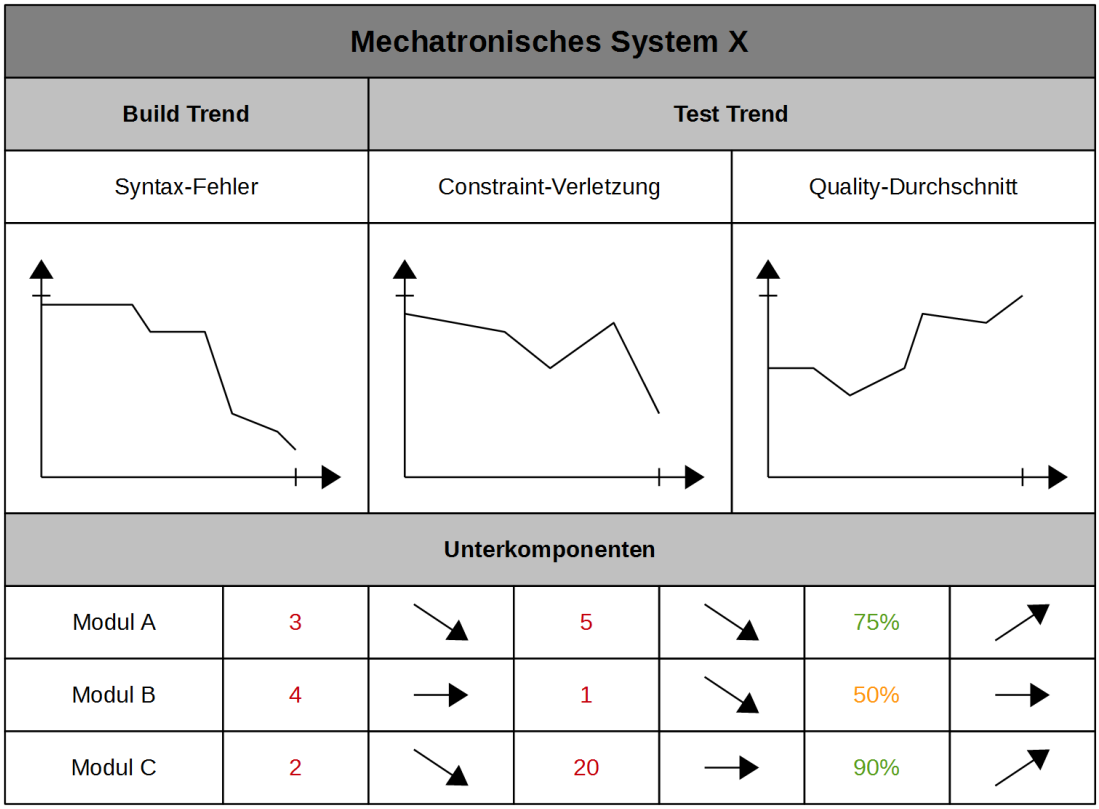


Abbildung 28: Entwurf der Benutzerschnittstelle für den Zugriff auf die Datenbank der kontinuierlichen Integration

Grundsätzlich orientiert sich die Darstellung an der Komponentenhierarchie des Systemmodells, um sowohl einen Überblick über das Gesamtsystem als auch einen detaillierten Einblick in die einzelnen Module bis hin zu atomaren Komponenten zu ermöglichen. Für die einzelnen Komponenten relevante Trends sind dabei die Anzahl der syntaktischen und semantischen Fehler im entsprechenden Komponenten-teilbaum, welche über den Verlauf der Entwicklung idealerweise kontinuierlich abnehmen. Ein weiterer wichtiger Faktor ist die Entwicklung der Qualitäten, d.h. der

numerischen Bewertung der einzelnen Systemzustände über die ausgeführten Simulationsläufe. Wie bereits zuvor erwähnt, ist aufgrund der Anzahl der Einzelwerte (abhängig von der Anzahl der modellierten Qualitäten sowie der Anzahl und Länge der Simulationsläufe) eine Aggregation notwendig. Grundsätzlich bieten sich verschiedene statistische Aggregationsoperatoren wie das Minimum oder Maximum sowie der Durchschnitt über die Einzelbewertungen an. Es wird an dieser Stelle jedoch davon ausgegangen, dass der Entwickler die gewünschte Form der Aggregation angeben muss, um möglichst aussagekräftige Werte zu erhalten. Die Modellierung dieser Aggregationen wurde im Projekt IMoMeSA nicht weiter betrachtet und ist Teil künftiger Forschungsarbeiten.

3.1.3 Workflow

Um sowohl die Modellierungstechnik als auch die eben vorgestellten Qualitätssicherungsmaßnahmen in einem mechatronischen Entwicklungsprozess zielführend anwenden zu können, bedarf es eines systematischen Vorgehens bzw. eines adäquaten Workflows. Vor diesem Hintergrund wurde im Rahmen von IMoMeSA ein generischer Entwicklungs-Workflow für die Phase der Konzeption erarbeitet, der einzelne Schritte zum Aufbau von Systemmodellen definiert und in eine zeitliche Abfolge bringt. Eine Übersicht des erarbeiteten Vorgehens ist in Abbildung 29 gezeigt, wobei der übergeordnete Ablauf gemäß Abbildung 9 zur besseren Einordnung erneut aufgegriffen ist.

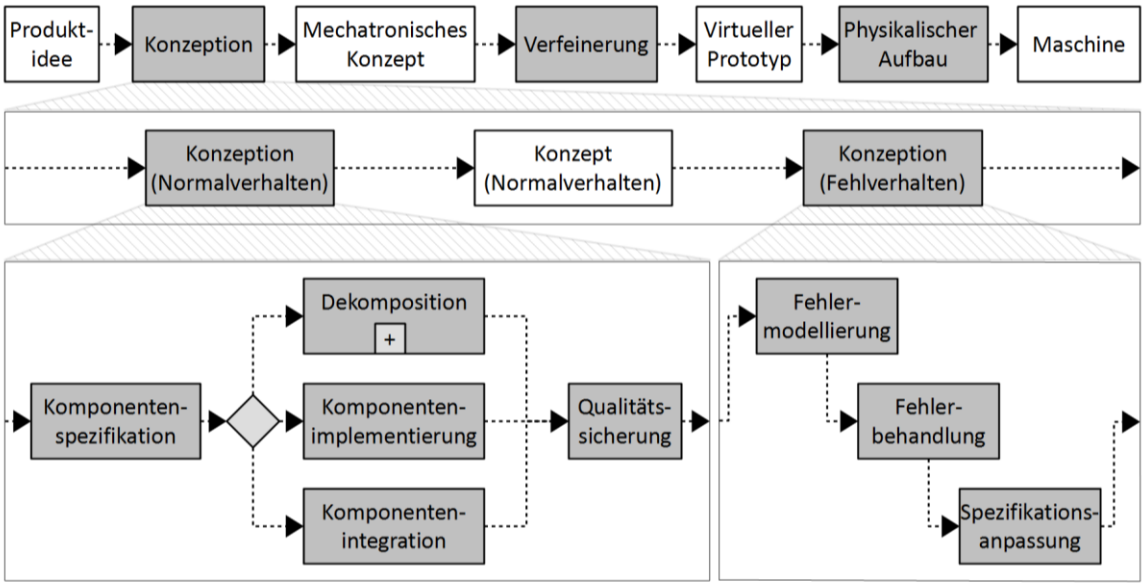


Abbildung 29: Übersicht über das systematische Vorgehen bei der Konzeption mechatronischer Systeme nach dem IMoMeSA-Ansatz

Grundsätzlich schlägt der IMoMeSA-Ansatz ein zweistufiges Vorgehen bei der Konzeption mechatronischer Systeme vor, bei dem zunächst ausschließlich das Gut- bzw. Normalverhalten aller beteiligten Komponenten betrachtet und somit die Komplexität des zu entwickelnden Systems reduziert wird. Sobald ein abgesichertes und in sich stimmiges Konzept des Normalverhaltens der Maschine entwickelt wurde, kann mögliches Fehlverhalten wie der Ausfall von Sensoren und Aktoren in Betracht gezogen werden. Im zweiten Schritt der Konzeption wird somit das ursprüngliche Modell einer Maschine mit Normalverhalten in ein Modell eines robusten Systems

unter Berücksichtigung von möglichem Fehlverhalten transformiert. In den beiden nachfolgenden Abschnitten werden diese beiden Schritte (Normal- und Fehlverhalten) mit den jeweils durchzuführenden Aktivitäten detailliert betrachtet.

3.1.3.1 Normalverhalten

Die Konzeption des Normalverhaltens geht von der Produktidee aus und hat das Ziel, ein mechatronisches Modell des Systems unter ausschließlicher Berücksichtigung des Gut- bzw. Normalverhaltens von Komponenten zu entwickeln. Der erste Schritt der Konzeption des Normalverhaltens ist die Spezifikation des zu entwickelnden Systems. Diese Spezifikation umfasst dabei die Modellierung von Anforderungen, Ports, Eigenschaften, Szenarien und Monitoren. Anschließend ist die Entscheidung zu treffen, ob eine Zerlegung in Unterkomponenten notwendig ist oder nicht. Diese Entscheidung orientiert sich in der Regel an der Komplexität des zu entwickelnden Systems. Je komplexer ein System dabei ist, desto eher ist auch eine Zerlegung in Unterkomponenten sinnvoll. Sofern eine Zerlegung notwendig ist, schließt sich der Schritt der Dekomposition an. Dabei werden zunächst die Unterkomponenten des zu entwickelnden Systems definiert. Danach wird für jede der abgeleiteten Komponenten der bereits vorgestellte Workflow rekursiv wiederholt. Dies kann insbesondere zu weiteren Zerlegungen führen, wodurch beliebige Dekompositionsstufen erreicht werden können. Sobald mittels dieses Top-Down-Ansatzes eine Ebene erreicht wurde, an der ausschließlich Komponenten angelegt wurden, die nicht weiter zu zerlegen sind (= atomare Komponenten), können diese mit einem Verhalten oder geometrischen Bauteilen implementiert werden. Sobald auch dieser Schritt abgeschlossen ist, können diese Komponenten mittels eines Bottom-Up-Ansatzes auf der nächst höheren Hierarchieebene integriert werden. Dieser Schritt beinhaltet zunächst die Verknüpfung der implementierten Unterkomponenten mit entsprechenden Kanälen. Es kann aber auch in dieser Phase dazu kommen, dass zur Integration entsprechender Unterkomponenten neue Komponenten dem System hinzugefügt werden, zum Beispiel wenn zwei Steuerungskomponenten nur mittels einer zusätzlichen Gesamtsteuerung auf nächst höherer Ebene miteinander kommunizieren können oder wenn mechanische Bauteile zur Montage eine gemeinsame Grundplatte benötigen.

Sobald die Implementierung einer einzelnen atomaren Komponente abgeschlossen wurde oder mehrere Komponenten integriert wurden, bietet sich die Absicherung des Zwischenergebnisses mittels Simulation und Unit-Testen an. Die Mechanismen Kompilierung und kontinuierliche Integration kommen hingegen soweit möglich während der gesamten Konzeption des Normalverhaltens zum Einsatz. Sobald die syntaktische und semantische Korrektheit der Komponenten und des integrierten Systems nachgewiesen werden kann, ist der Schritt der Konzeption des Normalverhaltens abgeschlossen.

3.1.3.2 Fehlverhalten

Aufbauend auf dem mechatronischen Konzept hinsichtlich des Normalverhaltens wird in der Konzeption möglichen Fehlverhaltens das grundsätzliche Ziel verfolgt, dieses Konzept in seiner Robustheit zu steigern und gegen mögliche Verhaltensfehler abzusichern. Grundlage dieser Systematik zur Behandlung von Fehlern ist eine klare Terminologie des Begriffs „Fehler“ sowie dessen Bezug zur bisher vorgestellten IMoMeSA-Terminologie, wie in Abbildung 30 dargestellt.

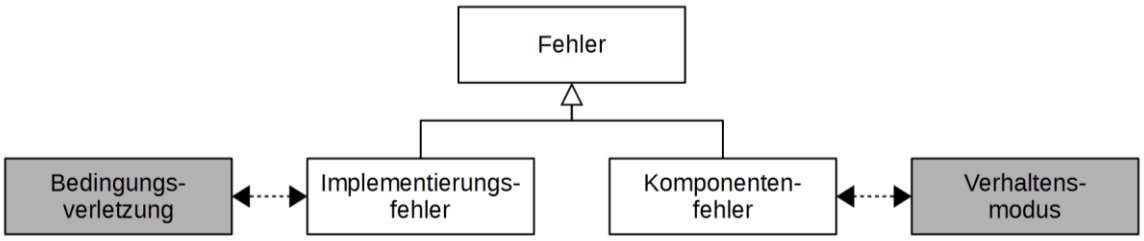


Abbildung 30: Terminologie des Begriffs „Fehler“ im Sinne von Implementierungs- und Komponentenfehlern

Prinzipiell werden zwei Arten von Fehlern unterschieden: Implementierungsfehler und Komponentenfehler. Erstere meinen, dass das modellierte mechatronische System in mindestens einem der Testfälle eine Bedingung verletzt und somit nicht akzeptiert werden kann. Komponentenfehler meinen auf der anderen Seite, dass eine Komponente in einem speziellen Fehlermodus agiert und somit das Verhalten der Komponente vom Normalverhalten abweicht. Prinzipiell ist festzuhalten, dass ein Komponentenfehler nicht automatisch einen Implementierungsfehler nach sich zieht. Vielmehr kann ein Komponentenfehler durch geeignete Mechanismen erkannt und behandelt werden, was gleichzeitig dem wesentlichen Ziel dieses Entwicklungsschrittes entspricht.

Der erste Schritt innerhalb dieses Vorgehens sieht zunächst die Modellierung von Komponentenfehlern vor. Diese können zum Beispiel das Ausfallen eines Sensors oder ein Kabel- bzw. Achsenbruch sein und werden über entsprechende Ereignis-Ports und Verhaltensmodi mit entsprechenden Zustandsautomaten beschrieben. Des Weiteren können relevante Komponentenfehler auf zwei Arten identifiziert werden: Einerseits können die atomaren Komponenten (z.B. Sensoren und Aktoren) eines mechatronischen Systemmodells direkt als Ausgangspunkt für die Modellierung von Komponentenfehlern dienen. Dabei steht für jede der atomaren Komponenten die Frage im Zentrum, welche Verhaltensmodi neben dem Normalverhalten noch möglich sind. Andererseits können die Bedingungen (vgl. Abschnitt 3.1.1.4) des mechatronischen Systemmodells als Ausgangspunkt für die Identifikation relevanter Komponentenfehler dienen. Dabei steht die Frage im Zentrum, was passieren muss, damit letztendlich eine entsprechende Bedingung verletzt wird. In beiden Fällen werden schließlich entsprechende Ereignis-Ports und Fehlermodi in das Modell integriert. Zudem müssen Fehlerszenarien aus bestehenden Szenarien durch gezielte Einbringung von Fehlereignissen abgeleitet werden, sodass die Effekte der Fehlermodi analysiert werden können. In der Regel können bei der Simulation der Fehlerszenarien bereits Implementierungsfehler (d.h. semantische Fehler) festgestellt werden, da das Verhalten der Komponenten auf den Gut- bzw. Normalfall ausgelegt war und unter Fehlerumständen in der Regel zu Bedingungsverletzungen führt.

Der zweite Schritt zur Konzeption eines robusten mechatronischen Systems sieht deshalb die Behandlung entstehender Implementierungsfehler vor. Dazu gliedert sich der Schritt der Fehlerbehandlung (vgl. Abbildung 29) in entsprechende Teilschritte, die in Abbildung 31 dargestellt sind.

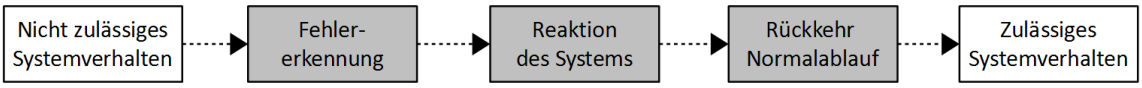


Abbildung 31: Teilschritte zur Behandlung von Implementierungsfehlern als Resultat eingebrachter Komponentenfehler

Die Fehlerbehandlung muss zunächst mit der Erkennung des vorliegenden Komponentenfehlers bzw. Fehlermodus beginnen. Dazu müssen in der Regel entsprechende Datensignale sinnvoll ausgewertet werden. Ein möglicher Ansatz dazu wird in Abschnitt 3.1.4 beschrieben. Nach der Erkennung des vorliegenden Komponentenfehlers kann eine entsprechende Reaktion des Systems eingeleitet werden. Eine mögliche Reaktion kann zum Beispiel das Wiederholen eines Teilablaufs oder das Anhalten der Motoren und die Meldung an den Maschinenbediener sein. Nachdem auf den Fehler geeignet reagiert wurde, kann die Rückkehr in den Normalablauf eingeleitet werden. Diese kann entweder vollautomatisch oder mit Eingriff des Bedieners erfolgen. Eine vollautomatische Rückkehr ist zum Beispiel möglich, wenn eine temporäre Verklemmung durch das Wiederholen einer Aktion aufgelöst werden kann. Der Eingriff des Bedieners ist hingegen erforderlich, wenn zum Beispiel defekte Komponenten (d.h. Komponenten, die sich in einem Fehlermodus befinden) ausgetauscht werden müssen, bevor der Normalablauf wieder aufgenommen werden kann.

Der letzte Schritt der Konzeption von mechatronischen Systemen umfasst schließlich die Anpassung der Spezifikation, d.h. insbesondere der Fehlerszenarien und Monitore. Grund dafür ist, dass sich durch die Betrachtung von Komponentenfehlern die Anforderungen an das System ändern können. Zum Beispiel kann der Prozess der Werkstückbearbeitung nun abhängig von den vorliegenden Fehlermodi entweder mit einem bearbeiteten Werkstück oder einer Fehlermeldung enden, während zuvor nur ein bearbeitetes Werkstück als Ergebnis akzeptiert wurde. Nach Spezifikations-anpassung sollten durch die Mechanismen der Qualitätssicherung keine syntaktischen und semantischen Fehler mehr vorliegen. Dies gilt sowohl für die ursprünglichen Szenarien als auch für die während der Konzeption des Fehlverhaltens hinzugefügten Fehlerszenarien.

3.1.4 Fehlerdiagnose

Um den Vorgang der Fehlererkennung (vgl. Abbildung 31) zu unterstützen, wurde im Rahmen des Projektes IMoMeSA bzw. der Konzeptionsphase des modellbasierten Entwicklungsprozesses noch die Möglichkeit betrachtet, Mechanismen zur Fehlerdiagnose aus dem Modell des mechatronischen Systems systematisch abzuleiten. Unter Fehlerdiagnose ist in diesem Kontext das Erkennen von Komponentenfehlern auf Basis von verfügbaren Datensignalen zu verstehen. Dadurch bildet sie die Grundlage für eine anschließende Fehlerbehandlung. Die gewählte Systematik für die Entwicklung von Fehlerdiagnosemechanismen ist schließlich in Abbildung 32 skizziert.

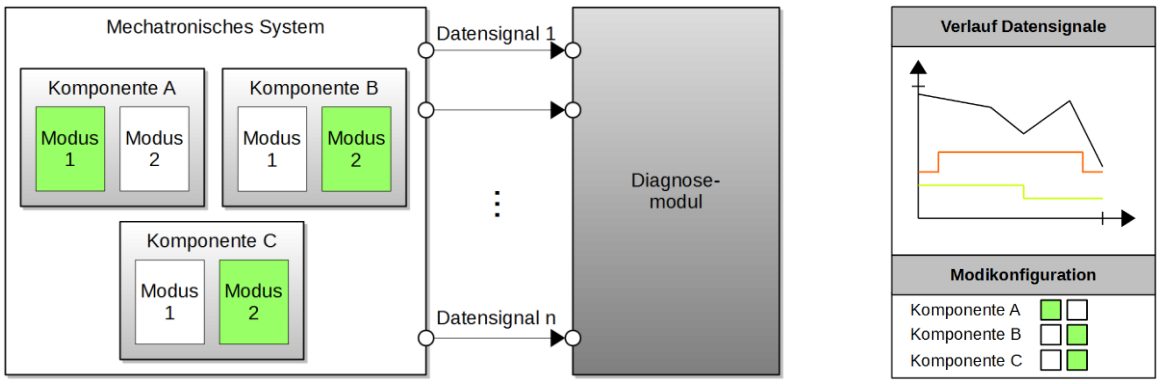


Abbildung 32: Diagnosemodul zur Aufzeichnung von Datensignalen während der Simulation bei einer bestimmten Modi-Konfiguration

Die Systematik sieht es vor, dass relevante Fehlerkonfigurationen zunächst über Fehlerszenarien explizit simuliert werden. In der Abbildung wurden zum Beispiel Modus 1 für Komponente A und Modus 2 für die Komponenten B und C gewählt. Während eines Simulationslaufes werden die resultierenden Signalverläufe, die später auch im Betrieb der Maschine zur Verfügung stehen, errechnet. Die Signalverläufe zeigen zum Beispiel, dass diverse Sensorwerte nicht wie erwartet eintreffen oder die Ansteuerung eines Aktors nicht den gewünschten Effekt liefert. Die Menge der errechneten Signalverläufe wird schließlich inklusive der zugehörigen Fehlerszenarien bzw. -konfigurationen in einer Datenbank für die künftige Verarbeitung gesammelt. Je mehr Fehlerszenarien dabei abgedeckt sind, desto vollständiger ist das Wissen über mögliche Effekte von Komponentenfehlern in bestimmten Situationen. Nach Aufbau der Datenbank werden Verfahren des maschinellen Lernens angewandt, um den Zusammenhang zwischen Signalverläufen und Fehlerkonfigurationen zu erlernen. Je eindeutiger dabei gewisse Datensignale oder Kombinationen von Datensignalen auf eine Fehlerkonfiguration hinweisen, desto genauer kann die erlernte Klassifikation die vorliegende Fehlerkonfiguration erkennen. Dieses Wissen kann anschließend genutzt werden, um entsprechende Fehlerbehandlungsroutinen zielgerichtet abzuleiten und zu implementieren.

Welche Verfahren des maschinellen Lernens dafür besonders gut geeignet sind (z.B. Support Vector Machines [36], Random Forrests [37] oder künstliche neuronale Netze [38]) und wie gut die Systematik in der Praxis funktioniert, wurde an dieser Stelle nicht weiter betrachtet. Vielmehr war es Ziel des Projektes, einen denkbaren Weg ausblickend aufzuzeigen.

3.2 Verfeinerung

Nach der Ausarbeitung eines mechatronischen Konzeptes mittels des in der Konzeption vorgestellten Ansatzes wird dieses in der Verfeinerung nun um Detailinformationen aus einzelnen Disziplinen angereichert, um schließlich einen virtuellen Prototyp zu erhalten (vgl. Abbildung 9). Um die Anreicherung des Modells zu ermöglichen, werden sowohl die Modellierungstechnik (vgl. Abschnitt 3.1.1) als auch die Mechanismen der Qualitätssicherung (vgl. Abschnitt 3.1.2) erweitert, sodass insbesondere physikalische Eigenschaften und Interaktionen genauer beschrieben werden können. Eine erste Übersicht über diese Erweiterungen bietet Abbildung 33.

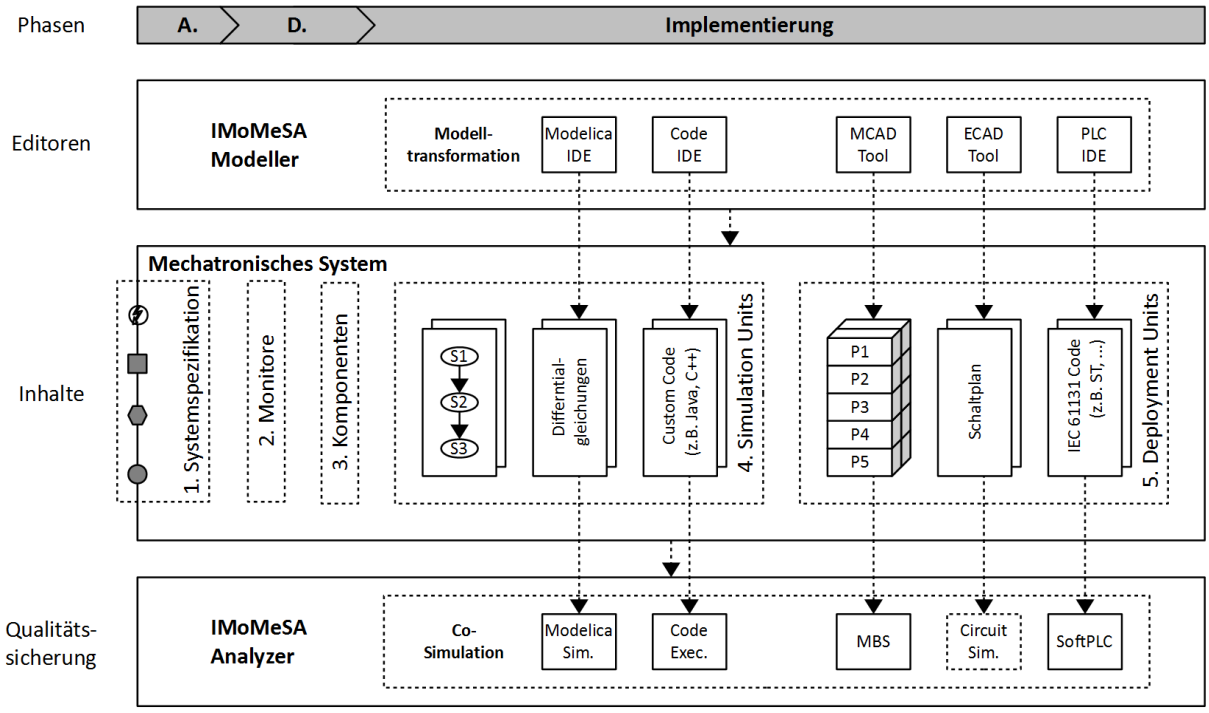


Abbildung 33: Übersicht über den Schritt der Verfeinerung, bestehend aus Phasen, Editoren, den Inhalten sowie der Qualitätssicherung

Prinzipiell umfasst der Ansatz für die Verfeinerung ebenfalls die drei Phasen Analyse, Design und Implementierung des Systems Engineerings [22]. Für die ersten beiden Phasen werden die bereits eingeführten Elemente der Modellierungstechnik (d.h. Anforderungen, Ports, Eigenschaften, Szenarien, Monitore und Komponenten bzw. Kanäle) unverändert weiter verwendet. In der Phase der Implementierung kommen jedoch neue Modellierungselemente hinzu, die entsprechende Erweiterungen der Editoren und der Qualitätssicherung nach sich ziehen. Für die Bearbeitung der Modelle kommen insbesondere bereits etablierte Werkzeuge des Werkzeugmaschinenbaus (vgl. Abschnitt 2.2.2) zum Einsatz, wobei Modelltransformationen einen nahtlosen Übergang gewährleisten. Für die Qualitätssicherung wird hingegen eine standardisierte Lösung für die Realisierung einer Co-Simulation integriert, um die zuvor eingeführten Mechanismen weiterhin nutzen zu können. Im Folgenden werden zunächst die Erweiterungen der Modellierungstechnik vorgestellt (Abschnitt 3.2.1), bevor die bereitgestellten Modelltransformationen erläutert werden (Abschnitt 3.2.2). Daraufhin wird auf die standardisierte Plattform für die Co-Simulation eingegangen (Abschnitt 3.2.3), bevor ein systematisches Vorgehen bei der Verfeinerung vorgeschlagen wird, das die Anwendung der Techniken erleichtern soll (Abschnitt 3.2.4).

3.2.1 Modellierungstechnik

Wie bereits erwähnt, beziehen sich die Erweiterungen der Modellierungstechnik für die Verfeinerung hauptsächlich auf Elemente, welche für die Beschreibung der Implementierung eingesetzt werden. Dabei unterscheidet der IMoMeSA-Ansatz zwischen zwei Kategorien von Elementen: Simulation und Deployment Units. Erstere werden in das Modell des mechatronischen Systems integriert, um weiterführende physikalische Effekte (z.B. Schwingungen oder Steifigkeiten) simulieren und damit in der Qualitätssicherung betrachten zu können. Simulation Units haben jedoch in der Regel für den physikalischen Aufbau der Maschine (d.h. die Beschaffung, Fertigung, Montage und Inbetriebnahme) keine direkte Bedeutung. Deployment Units auf der anderen Seite beschreiben genau diejenigen Modellaspekte der Maschine, die für den physikalischen Aufbau relevant sind. Darunter fallen die mechanische und elektrische CAD-Konstruktion sowie die Steuerungssoftware. Im Folgenden werden die einzelnen Simulation und Deployment Units im Detail beschrieben.

3.2.1.1 Simulation Units

Um die physikalischen Zustände und das daraus resultierende Zeitverhalten mechatronischer Systeme genauer beschreiben zu können, bietet der IMoMeSA-Ansatz in der Phase der Verfeinerung drei neue Mechanismen: Hybride Ein-/Ausgabe-Automaten [39], Differenzialgleichungen im Modelica-Format [40] sowie proprietären Programmcode. Im Folgenden werden die Mechanismen genauer erläutert.

Hybride Ein-/Ausgabe-Automaten

Während der Konzeption kommen klassische Eingabe/Ausgabe-Automaten zum Einsatz, um das zeitdiskrete Verhalten der Komponenten zu beschreiben (vgl. Abschnitt 3.1.1.8). Hybride Eingabe/Ausgabe-Automaten bieten demgegenüber den Vorteil, dass auch zeitkontinuierliche Ein- und Ausgabeströme modelliert werden können. Als Grundlage dafür wird die Modellierungstechnik um die Modellierung von Differenzialgleichungen auf numerischen Variablen und Ausgabe-Ports erweitert. Damit kann zum Beispiel der Zusammenhang zwischen Weg und Geschwindigkeit deutlicher genauer beschrieben werden, wie auch Abbildung 34 zeigt.

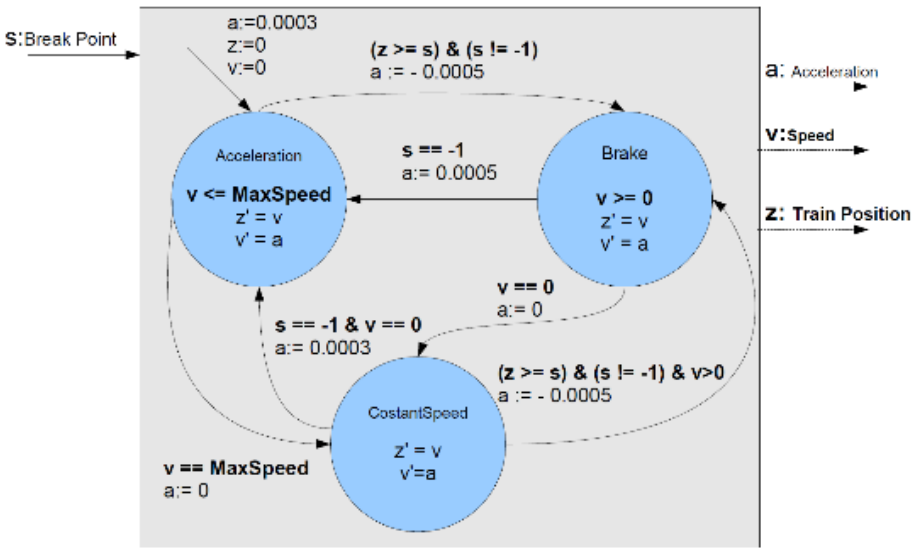


Abbildung 34: Beispiel eines hybriden Eingabe/Ausgabe-Automaten [39]

Wie in dem Beispiel zu sehen ist, werden Differenzialgleichungen an die Zustände des hybriden Automaten gehängt. Konkret ist im Zustand „Acceleration“ die Geschwindigkeit als Ableitung des Weges und die Beschleunigung als Ableitung der Geschwindigkeit definiert. Zudem ist an dieser Stelle auch eine Bedingung (vgl. Abschnitt 3.1.1.4) definiert, dass die Geschwindigkeit einen gewissen Schwellenwert nicht überschreiten darf. Zustandsübergänge hingegen sind ebenfalls über Wächter und Aktionen beschrieben. Die Ausdrücke der Wächter und Aktionen können dabei sowohl auf die Werte der kontinuierlichen als auch der diskreten Ströme zugreifen. Für weiterführende Informationen zu hybriden Eingabe/Ausgabe-Automaten und deren Simulation sei an dieser Stelle auf [39] verwiesen.

Modelica-Modelle

Hybride Ein-/Ausgabe-Automaten erwarten, dass für jeden zeitkontinuierlichen Strom die Richtung angegeben wird. In manchen Fällen kann es jedoch schwierig sein, die Richtung eines zeitkontinuierlichen Wertestroms festzulegen, wie z.B. bei der Modellierung von elektrischen Netzen mit Spannungsquellen, Übertragungswegen und Widerständen. Die Richtung der Stromflüsse ergibt sich dabei dynamisch durch die konkrete Netztopologie sowie den aktuellen Zustand der einzelnen Komponenten. Um die Modellierung solcher Phänomene zu vereinfachen, bietet der IMoMeSA-Ansatz die Möglichkeit, DGL-Systeme über Komponentengrenzen hinweg nach dem Modelica-Vorbild zu modellieren. Ein Beispiel dazu ist in Abbildung 35 gezeigt.

```
1  connector Pin
2     Voltage v;
3  end Pin;
4
5  model Capacitor
6     Pin a;
7  Equation
8     der(a.v) = 1;
9  end Capacitor;
```

Abbildung 35: Beispiel eines Modelica-Modells [40] für die Modellierung zeitkontinuierlichen Verhaltens

Der Kerngedanke von Modelica [40] ist ebenfalls eine komponentenbasierte Modellierung mit Konnektoren (entsprechen den Ports im IMoMeSA-Ansatz) sowie Variablen und Gleichungen. Variablen beschreiben dabei die (physikalischen) Zustände der Komponenten und Konnektoren, während Gleichungen diese Zustände zueinander in Beziehung setzen. Die Gleichungen können insbesondere die Ableitung einzelner Variablen nach der Zeit enthalten, wodurch das Zeitverhalten des Systems beschrieben wird. Durch Verbindung von Konnektoren lassen sich des Weiteren die Zustände verschiedener Komponenten in Verbindung setzen. Dieser Mechanismus erzeugt letztlich ein großes Differenzialgleichungssystem, welches für die Simulation gelöst werden muss. Es sei an dieser Stelle angemerkt, dass eine solche Lösung für hybride Eingabe/Ausgabe-Automaten nicht berechnet werden muss, was die Implementierung der Simulation vereinfacht.

Proprietärer Programmcode

Falls weder hybride Eingabe/Ausgabe-Automaten noch Modelica-Modelle geeignet sind, um ein gewünschtes Verhalten zu modellieren, bietet der IMoMeSA-Ansatz die Möglichkeit, proprietären Programmcode in das Modell des mechatronischen Systems zu integrieren. Ein typischer Anwendungsfall ist zum Beispiel die Anbindung eines Ray Tracers [41] für das Synthetisieren realistischer Kamerabilder, um Bilderkennungsalgorithmen über die Mechanismen der Qualitätssicherung testen zu können. Ein Beispiel für proprietären Programmcode ist in Abbildung 36 gezeigt.

```
1  class MyCustomCode extends CustomCode
2  {
3      public void execute(int time)
4      {
5          // Read port
6          Object value = getComponent().getPort("<input>").get(time);
7          // Write port
8          getComponent().getPort("<output>").set(time, ...);
9      }
10 }
```

Abbildung 36: Beispiel für proprietären Programmcode in der Sprache Java [33]

Wie im Beispiel gezeigt, wird im proprietären Programmcode in erster Linie eine erweiterte Methode für das Ausführen der Simulation implementiert. Diese erwartet als Parameter den Zeitpunkt, für welchen die Berechnung stattfinden soll. Weitere Parameter und Rückgabewerte werden nicht benutzt. Stattdessen können Variablen und Ports innerhalb der Methode gelesen und geschrieben werden. Der Zugriff erfolgt dabei über die Komponente, für die der proprietäre Programmcode definiert ist. Es ist an dieser Stelle zu erwähnen, dass der Programmcode uneingeschränkte Berechnungen vornehmen kann. So sind sowohl der Zugriff auf das Netzwerk als auch der Aufruf anderer Programme über die Kommandozeile denkbar. Jedoch sollte die benötigte Zeit für die Berechnung berücksichtigt werden, da die Simulation so lange blockiert ist, bis das Ergebnis zurückgegeben wurde.

3.2.1.2 Deployment Units

Neben den Simulation Units bietet der IMoMeSA-Ansatz sogenannte Deployment Units für die Beschreibung derjenigen Modellierungsaspekte an, die für den physikalischen Aufbau direkt relevant sind. Konkret wird die Modellierungstechnik an dieser Stelle um computergestützte Modelle der Mechanik (MCAD) und der Elektrik bzw. Elektronik (ECAD) sowie um Steuerungsprojekte nach dem Standard IEC 61131-3 erweitert. Im Folgenden werden diese Elemente genauer beschrieben.

MCAD-Modelle

Die mechanische Konstruktion wird im Werkzeugmaschinebau heute in der Regel mit sogenannten MCAD-Werkzeugen gestaltet. Ziel des IMoMeSA-Projektes war es, diese Werkzeuge nahtlos in den beschriebenen Ansatz zu integrieren und somit von deren großem Funktionsumfang sowie dem vorhandenen Mitarbeiter-Knowhow in den Unternehmen zu profitieren. Um diese Integration zu ermöglichen, wurde die Modellierungstechnik in der Verfeinerung um ein standardisiertes Format für die Beschreibung von MCAD-Modellen erweitert. Konkret handelt es sich um das Format COLLADA [42], dessen Aufbau schematisch in Abbildung 37 gezeigt ist.

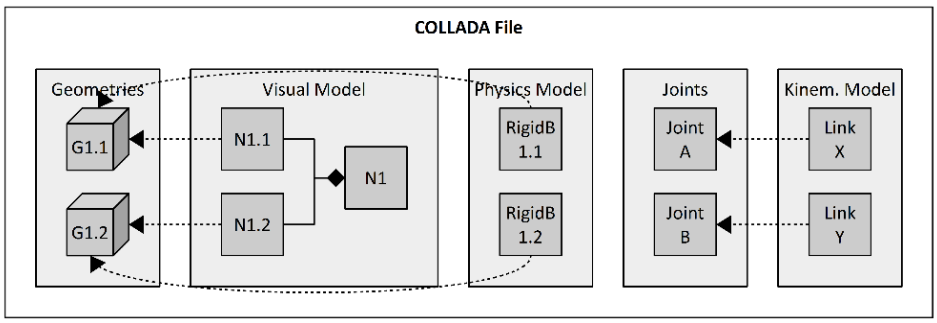


Abbildung 37: Schematischer Aufbau des standardisierten Formats COLLADA [42]

Das COLLADA-Format stellt unterschiedliche Bereiche für die Beschreibung einzelner Aspekte des MCAD-Modells zur Verfügung. Kern der Modelle ist eine sogenannte Geometriebibliothek, welche die Geometrien der Bauteile und Material-Ports speichern kann. Darauf aufbauend definiert das visuelle Modell eine Knotenhierarchie, welche die Komponentenstruktur des mechatronischen Systems widerspiegeln kann. Die Knoten der Hierarchie definieren dabei jeweils ein lokales Koordinatensystem, in welchem die Geometrien durch entsprechende Transformationen verankert werden. Physikalische Aspekte werden des Weiteren in einem Physikmodell abgelegt. Dort werden insbesondere die Festkörper des mechanischen Aufbaus definiert. Die Definition von Festkörpern ist insbesondere für die Anbindung einer Simulation mittels Mehrkörper-System (MKS) sinnvoll. Kinematische Aspekte werden in einem kinematischen Modell festgelegt, welches auf einer Gelenkbibliothek (engl. Joint Library) aufbaut. Gelenke definieren zunächst die möglichen Freiheitsgrade einer Bewegung. Die Verbindungen des kinematischen Modells erzeugen Gelenkketten, welche zum Beispiel die Modellierung von Roboterarmen erlauben. Die Integration der MCAD-Modelle in die IMoMeSA-Modellierungstechnik sieht vor, dass für jede Komponente des IMoMeSA-Modells ebenfalls eine entsprechende COLLADA-Datei existiert, welche den gesamten Komponentenbaum (d.h. die Komponente selbst, deren Unterkomponenten, usw.) abbildet. Die Aufteilung der COLLADA-Dateien nach Komponenten und ein entsprechendes Referenzieren sind dabei möglich.

ECAD-Modelle

Ähnlich wie für die mechanische Konstruktion nutzen die Unternehmen des Werkzeugmaschinenbaus heute für die elektrische bzw. elektronische Konstruktion sog. ECAD-Werkzeuge. Auch bezüglich dieser war es das Ziel des Projektes IMoMeSA, die nahtlose Integration bestehender Werkzeuge in die modellbasierte Entwicklungsmethodik zu ermöglichen. Beispielhaft wurde dafür das Werkzeug EPLAN P8 ausgewählt, welches derzeit eine breite Anwendung und hohe Nutzer-akzeptanz findet. Der schematische Aufbau einer EPLAN-Datei für die Beschreibung der elektrischen bzw. elektronischen Konstruktion ist in Abbildung 38 gezeigt.

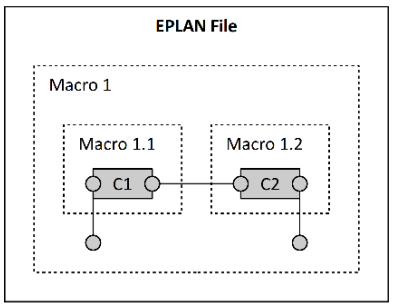


Abbildung 38: Schematischer Aufbau einer EPLAN-Datei

Das EPLAN-Format beschreibt hauptsächlich den geometrischen Aufbau der elektrischen bzw. elektronischen Konstruktion. Konkret werden Position und Orientierung von Komponenten sowie Ankerpunkte und Verbindungen definiert. Ankerpunkte erlauben den Zusammenschluss von Komponenten, während Verbindungen die notwendige Verdrahtung inklusive Verbindungseigenschaften wie Kabeldurchmesser und -material beschreiben. Die Modellierung des logischen bzw. funktionalen Aufbaus der Konstruktion wird schließlich durch Makros unterstützt. Makros sind ein allgemeiner Mechanismus zur Wiederverwendung von Teilkonstruktionen in ECAD-Dateien.

Die Integration der ECAD-Modelle mit dem IMoMeSA-Modellierungsansatz erfolgt schließlich dadurch, dass jeder IMoMeSA-Komponente genau ein EPLAN-Makro zugewiesen ist. Dieses Makro beschreibt den geometrischen Aufbau der elektrischen bzw. elektronischen Konstruktion der IMoMeSA-Komponente. Insbesondere werden dabei auch die Makros der definierten Unterkomponenten eingebunden, um auch hier eine Wiederverwendung zu ermöglichen.

IEC 61131-3-Code

Den letzten Baustein der Deployment Units neben den MCAD- und ECAD-Modellen bildet die Steuerungssoftware, welche üblicherweise während der Inbetriebnahme auf eine speicherprogrammierbare Steuerung (SPS) aufgespielt wird. Wie bei den vorigen Deployment Units sind auch hier etablierte Werkzeuge im Werkzeugmaschinenbau im Einsatz, welche die Entwicklung der Steuerungssoftware erlauben. Aus diesem Grund war es das Ziel des IMoMeSA-Projektes, die nahtlose Integration dieser Werkzeuge zu ermöglichen. Verwendet wurde dabei das Format IEC 61131-3 [43] für den zu implementierenden Programmcode sowie den Gesamtaufbau des Steuerungsprojektes. Dieser Aufbau ist schematisch in Abbildung 39 visualisiert.

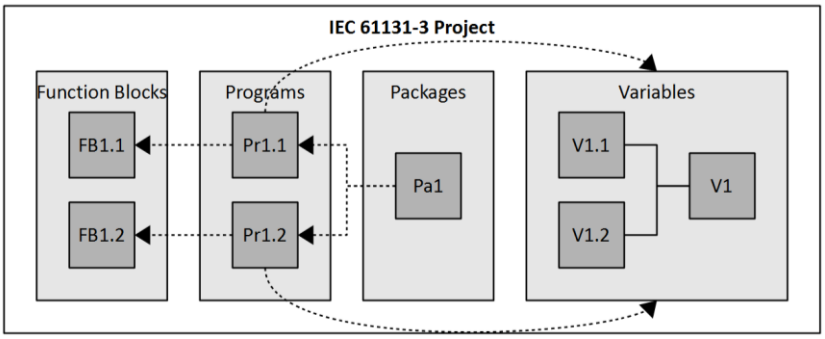


Abbildung 39: Schematischer Aufbau eines Projektes unter Nutzung des Formates IEC 61131-3

Die elementaren Bestandteile eines Steuerungsprojektes sind Funktionsblöcke, Programme und Variablen. Erstere definieren dabei wiederverwendbare funktionale Einheiten, die durch mehrere Eingabe- und Ausgabeparameter sowie eine ausführbare Logik gekennzeichnet sind. Programme definieren auf der anderen Seite die eigentlichen auszuführenden Einheiten der Steuerungssoftware. Intern können Programme beliebige Funktionsblöcke verwenden, um eine Strukturierung der Logik zu ermöglichen und somit letztlich die Qualität des Quellcodes (zum Beispiel im Sinne von Les- und Wartbarkeit) zu erhöhen. Des Weiteren können Programme auch auf globale Variablen zugreifen, um bestimmte Informationen zwischen-zu-speichern und mit anderen Programmen auszutauschen. Schließlich können zusätzlich Pakete verwendet werden, um Programme hierarchisch zu strukturieren und somit eine bessere Übersicht über die Steuerungssoftware zu erlangen.

Die Integration des Standards IEC 61131-3 mit der IMoMeSA-Modellierungstechnik erfolgt dadurch, dass für alle modellierten Softwarekomponenten ein Funktionsblock mit entsprechender Logik abgelegt wird. Des Weiteren existiert für das gesamte mechatronische Modell ein eigenes Projekt, welches über Pakete die mechatronische Komponentenstruktur abbildet.

3.2.2 Modelltransformationen

Um die nahtlose Integration der etablierten Werkzeuge im Werkzeugmaschinenbau mit den Deployment Units des IMoMeSA-Ansatzes zu ermöglichen, wurden Modelltransformationen ausgearbeitet, wodurch sich bestimmte Teile der MCAD- und ECAD-Modelle sowie Teile des IEC 61131-3 Codes automatisch erzeugen lassen. Im Folgenden werden die Transformationen für MCAD- und ECAD-Modelle sowie die automatische Generierung von Steuerungscode gemäß IEC 61131-3 beschrieben.

3.2.2.1 MCAD-Modelle

Um die Integration der MCAD-Dateien zu erleichtern, wurde für die Verfeinerung ein Weg gewählt, bei dem lediglich ausgewählte Teile des mechatronischen Modells vollständig in die MCAD-Dateien ausgelagert werden, während andere Teile durch Transformation vom IMoMeSA-Modell zum MCAD-Modell generiert werden. Abbildung 40 gibt eine Übersicht über die ausgelagerten Modellteile (weiß) und die generierten Modellteile (grau).

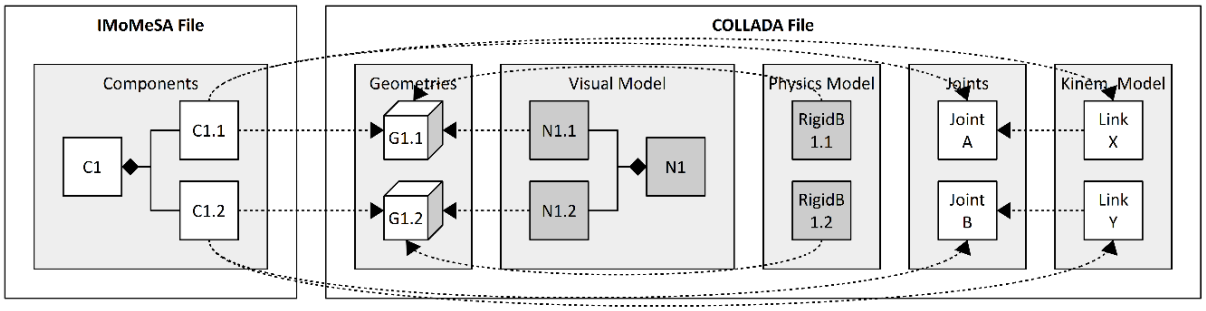


Abbildung 40: Schematische Darstellung der Integration des IMoMeSA-Modells mit dem MCAD-Modell

Wie die Abbildung zeigt, werden während der Verfeinerung die Geometrien (entsprechen den Bauteilen) sowie die Gelenke (entsprechen den kinematischen Energie-Ports) und die Verbindungen des kinematische Modells (entsprechen den

kinematischen Energiekanälen) in das MCAD-Modell ausgelagert. Der Grund dafür ist, dass die Auslagerung die Synchronisierung mit den MCAD-Werkzeugen erleichtert. Änderungen an den Geometrien oder den Achsen sind somit automatisch auch im IMoMeSA-Editor sichtbar. Eine derartige Synchronisation ist beim visuellen Modell nicht möglich. Der Grund dafür ist, dass das visuelle Modell in COLLADA die Komponentenhierarchie in IMoMeSA widerspiegelt. Die Komponentenhierarchie enthält jedoch deutlich mehr Informationen als das visuelle Modell. Daher wurde die Entscheidung getroffen, dass Änderungen an der Komponentenhierarchie ausschließlich im IMoMeSA-Editor (bzw. im IMoMeSA-File) vorgenommen werden dürfen und das visuelle Modell automatisch generiert wird. Ähnliches gilt für das physikalische Modell in COLLADA, wobei die Begründung darin liegt, dass jedes IMoMeSA-Bauteil automatisch auch ein COLLADA-Festkörper ist und somit keine explizite Modellierung in COLLADA notwendig ist.

3.2.2.2 ECAD-Modelle

Im Gegensatz zur eben vorgestellten Transformation von MCAD-Modellen werden bei der ECAD-Konstruktion während der Verfeinerung keine Modellteile ausgelagert, sondern ausschließlich Modelltransformationen genutzt, da die Dateiinhalte weitestgehend komplementär zueinander sind. Abbildung 41 gibt eine Übersicht über das IMoMeSA-Modell sowie das entsprechende EPLAN Modell.

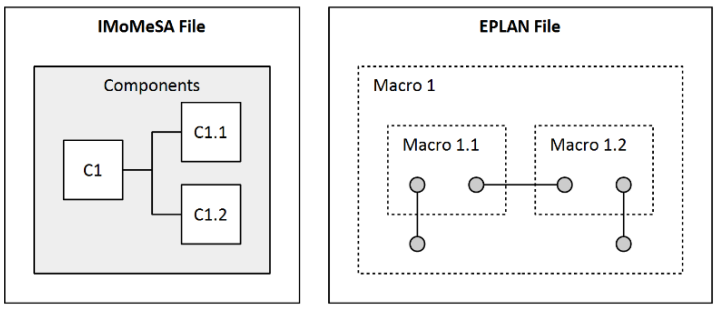


Abbildung 41: Schematische Darstellung der Integration des IMoMeSA-Modells mit dem ECAD-Modell

Wie bereits in Abschnitt 3.2.1.2 erwähnt, wird für jede IMoMeSA-Komponente zunächst ein EPLAN-Makro erzeugt, welches die elektrische bzw. elektronische Konstruktion der betrachteten Komponente enthalten soll. Dabei wird die Komponentenhierarchie durch entsprechende Beziehungen zwischen den Makros reflektiert. Des Weiteren wird für jeden elektrischen Energie-Port automatisch ein Ankerpunkt innerhalb der Makros erzeugt. Da die geometrische Position der Energie-Ports nicht im IMoMeSA-Modell enthalten ist, muss diese vom Elektrokonstrukteur manuell bestimmt werden. Zudem werden kinematische Energiekanäle des IMoMeSA-Modells in Verbindungen zwischen Ankerpunkten übersetzt. Auch hierbei kann der Kabeltyp nicht aus dem IMoMeSA-Modell ausgelesen werden. Er muss somit auf einen Standardwert initialisiert werden. Das ECAD-Werkzeug kann schließlich dazu verwendet werden, die einzelnen Makros inhaltlich auszufüllen. Hierbei sollte jedoch erwähnt werden, dass neue Komponenten, die auch in der Simulation betrachtet werden sollen, zunächst im IMoMeSA-Modell angelegt werden müssen, bevor eine entsprechende Repräsentation in dessen Makro aufgenommen wird. Sofern keine Simulation der Komponente geplant ist, reicht hingegen ausschließlich die Integration in dem entsprechenden Makro der Vaterkomponente.

3.2.2.3 IEC 61131-3-Code

Vergleichbar mit MCAD-Modellen wird bei Steuerungsprojekten während der Verfeinerung wiederum ein hybrider Ansatz genutzt, bei dem ausgewählte Modellteile vollständig in das IEC 61131-3-Projekt ausgelagert sind, während andere Modellteile automatisch generiert werden. Abbildung 42 gibt eine Übersicht über ausgelagerte Modellteile (weiß) und generierte Modellteile (grau).

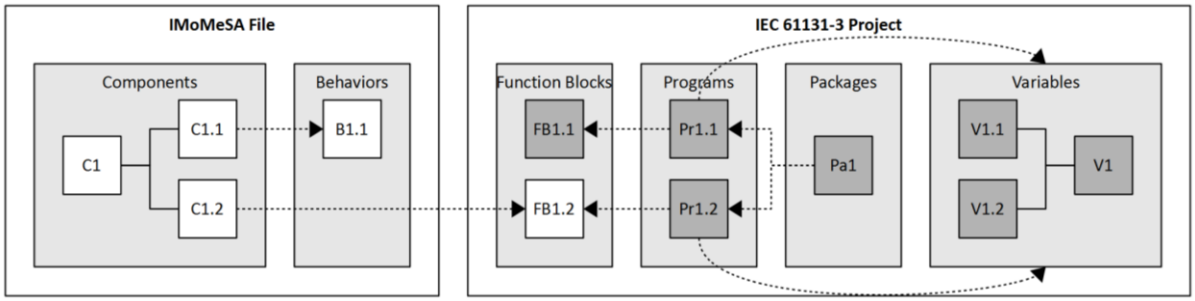


Abbildung 42: Schematische Darstellung der Integration des IMoMeSA-Modells mit dem IEC 61131-3-Projekt

Das Verhalten einzelner Softwarekomponenten kann während der Verfeinerung ausgelagert werden. Für so behandelte Komponenten ist keine Beschreibung mehr über Zustandsautomaten vorgesehen, sondern es können direkt die Programmiersprachen der IEC 61131-3 zur Implementierung einzelner Funktionsbausteine verwendet werden. Alternativ kann ein Funktionsbaustein aber auch vollständig aus den Verhaltensautomaten aus der Konzeption generiert werden, sofern die gewünschte Steuerungsfunktionalität für die betrachtete Komponente bereits vollständig mittels der Automaten erfasst werden konnte. Unabhängig davon, ob ein Funktionsbaustein automatisch generiert oder während der Verfeinerung manuell implementiert wurde, werden Programme benötigt, die diese Funktionsbausteine instanziieren. Diese wiederum können genauso wie die Pakete und die Variablen (vgl. Abschnitt 3.2.1.2) automatisch aus dem IMoMeSA-Modell generiert werden, da sie sich direkt aus der Komponentenstruktur bzw. aus Ports und Kanälen ergeben.

Bei der Entwicklung des Codegenerators wurde insbesondere darauf geachtet, dass der generierte Code lesbar ist. Die Lesbarkeit des Codes wurde als eine der kritischen Anforderungen identifiziert, welche einerseits im Stand der Technik bisher weniger Beachtung findet, andererseits aber notwendig ist, um manuelle Änderungen möglichst einfach innerhalb des generierten Codes integrieren zu können. Weitere Anforderungen neben der Lesbarkeit waren die Erhaltung der Funktionalität (d.h. des Eingabe-/Ausgabeverhaltens) und der Struktur (d.h. der Zustände und Transitionen) sowie der Konsistenz bei manuellen Änderungen [14]. Auf Basis dieser Anforderungen wurde schließlich prototypisch eine Modelltransformation mit der Transformationssprache XSL [44] entwickelt. Ausschnitte des generierten Codes werden exemplarisch in Kapitel 4 bei der Verfeinerung des gewählten Fallbeispiels vorgestellt.

3.2.3 Qualitätssicherung

Sobald Teile des mechatronischen Systemmodells mittels der beschriebenen Erweiterungen der Modellierungstechnik bzw. der zugehörigen Modelltransformationen verfeinert wurden, können die ursprünglichen Mechanismen der Qualitätssicherung bzw. insbesondere die Simulation nicht mehr direkt angewendet werden.

Grund dafür ist, dass die ursprüngliche Modellierungstechnik bzw. Simulation auf einem zeitdiskreten Verhaltensmodell aufbaut, während in der Verfeinerung auch zeitkontinuierliche Verhaltensaspekte betrachtet werden. Die Simulation Units basieren hauptsächlich auf Differenzialgleichungen und entsprechenden Lösungsverfahren [45]. Im Rahmen der Deployment Units hingegen wird für die Mechanik eine Mehrkörpersimulation [46], für die Elektrik eine Schaltkreissimulation [47] und für die Steuerungstechnik eine SoftSPS [48] benötigt. Für jeden dieser Aspekte existieren bereits Simulatoren, die seit langer Zeit im industriellen Umfeld eingesetzt werden. Aus diesem Grund wurde im Rahmen des Projektes die Entscheidung getroffen, die bestehenden Simulatoren so weit wie möglich zu nutzen und ggf. zu integrieren. Für die Integration der Simulatoren baut der IMoMeSA-Ansatz dabei auf dem Functional-Mockup-Interface-Standard (FMI) [49] auf, welcher eine Infrastruktur für die Kopplung verschiedenster Simulatoren und deren Co-Simulation bietet. Abbildung 43 zeigt eine Übersicht über den Aufbau einer FMI-Infrastruktur.

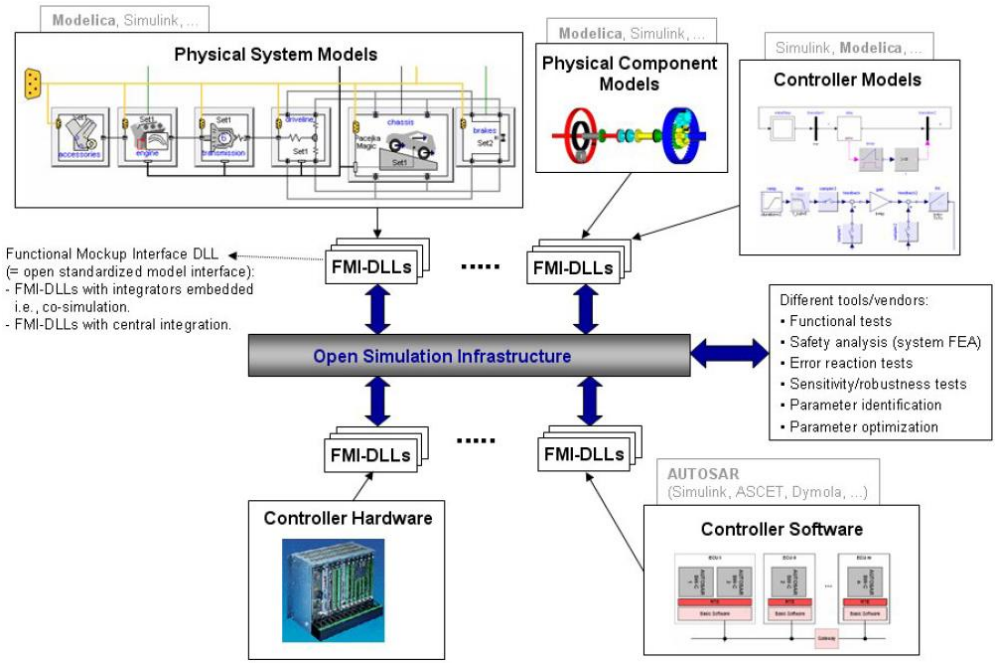


Abbildung 43: Infrastruktur des Functional-Mockup-Interface-Standards (FMI) [49]

Wie in der Abbildung dargestellt, bietet der FMI-Standard unter anderem die Möglichkeit, Modelica- [40] und Matlab-Simulink-Modelle [50] in einer Co-Simulation zu koppeln. Die einzelnen Simulatoren (Functional Mockup Units bzw. FMUs) werden dabei über eine sogenannte FMI-DLL an die Infrastruktur angebunden. Die FMI-DLL implementiert eine standardisierte Schnittstelle für den Aufruf des Simulators. Die Schnittstelle erlaubt unter anderem den Austausch von Zustandsinformationen zwischen FMUs sowie die Angabe des Zeithorizonts. Rückgabe der einzelnen FMUs sind die errechneten Zustände, wie auch Abbildung 44 visualisiert.

In IMoMeSA werden die FMUs bzw. FMI-DLLs konkret dafür verwendet, die Mehrkörpersimulation für die Mechanik, die Schaltkreissimulation für die Elektrik und SoftSPS für die Steuerungstechnik sowie die Simulatoren für hybride Eingabe/Ausgabe-Automaten, Modelica-Modelle und proprietären Code geeignet zu koppeln. Der Austausch von Zustandsinformationen ist dabei über die Schnittstellen der IMoMeSA-Komponenten definiert.

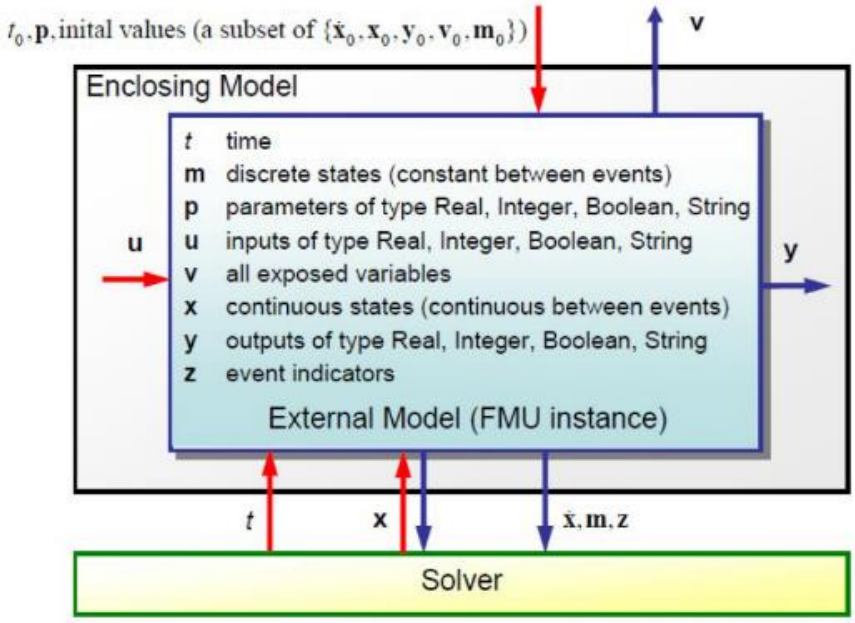


Abbildung 44: Schnittstelle von Functional Mockup Units (FMUs) [49]

3.2.4 Workflow

Analog zur Konzeption wurde auch für den Schritt der Verfeinerung ein systematisches Vorgehen definiert, wie ausgehend von einem mechatronischen Konzept ein funktionsfähiger virtueller Prototyp unter Verwendung der erweiterten Modellierungstechnik, der Modelltransformationen sowie des skizzierten Co-Simulationsansatzes erarbeitet werden kann. Grundsätzlich wird ein inkrementelles Vorgehen innerhalb der Verfeinerung vorgeschlagen, bei dem einzelne Komponenten sukzessive verfeinert werden (vgl. Abbildung 45). Dafür wird die Analyse des mechatronischen Systems als erster Schritt vorgeschlagen. Während dieses Schritts kann das mechatronische Konzept nach Komponenten durchsucht werden, die entweder nicht direkt umgesetzt werden können (z.B. aufgrund einer vereinfachten Geometrie) oder bei denen Zustandsautomaten das gewünschte Systemverhalten nur unzureichend abbilden. Als Ergebnis kann schließlich die Frage beantwortet werden, ob Komponenten existieren, für die eine Verfeinerung notwendig ist.

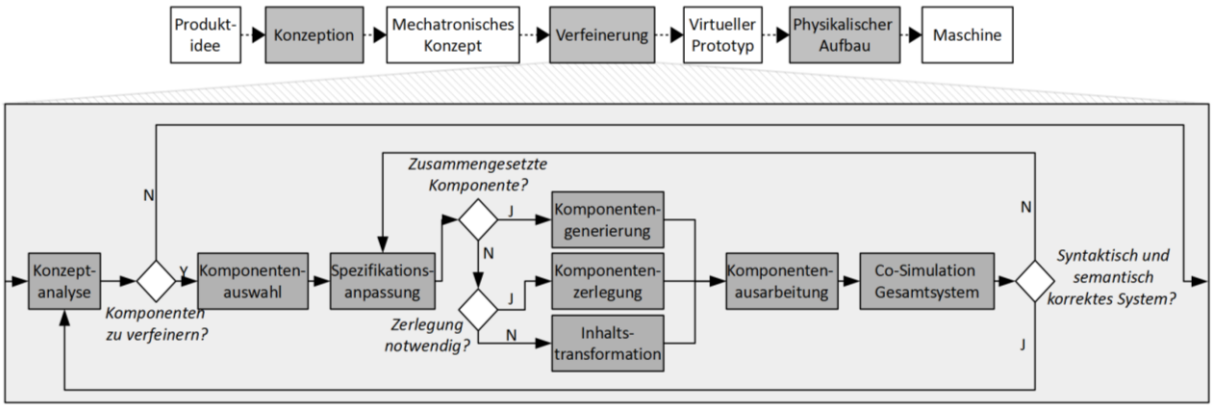


Abbildung 45: Übersicht über das systematische Vorgehen bei der Verfeinerung mechatronischer Systeme nach dem IMoMeSA-Ansatz

Sobald entsprechende Komponenten für die Verfeinerung identifiziert wurden, folgt der Schritt der Komponentenauswahl. An dieser Stelle wird die Entscheidung getroffen, welche der Komponenten aktuell bei der Verfeinerung Priorität hat. Diese Entscheidung kann sich zum Beispiel danach richten, welche Aspekte möglichst gut isoliert voneinander verfeinert werden können. Je besser sich ein Aspekt dabei isolieren lässt, desto einfacher sollte die Verfeinerung sein und desto schneller kann man sich um andere Aspekte kümmern. Ein weiteres Auswahlkriterium bildet die Unsicherheit bezüglich des gewählten Lösungsprinzips. Je größer diese Unsicherheit ist, desto sinnvoller ist eine zügige Betrachtung während der Verfeinerung, um mögliche Probleme so früh wie möglich zu identifizieren.

Für die ausgewählte Komponente muss anschließend (unabhängig davon, ob es sich um eine zusammengesetzte oder atomare Komponente handelt) die Spezifikation der Komponente angepasst werden. Diese Anpassung kann beispielsweise eine Erweiterung der Schnittstelle der Komponente oder die Modifizierung von Szenarien oder Monitoren beinhalten. Darauf aufbauend kann die Komponente selbst verfeinert werden. Dabei kann grundsätzlich zwischen drei Varianten unterschieden werden, die vom Typ der zu verfeinernden Komponente abhängen. Wenn die betrachtete Komponente bereits aus Unterkomponenten zusammengesetzt ist, bedeutet die Aufgabe der Verfeinerung die Generierung von neuen Komponenten innerhalb der bestehenden Struktur der zugehörigen Unterkomponenten und die anschließende Ausarbeitung unter Verwendung von Simulation oder Deployment Units. Ein Beispiel für diesen Fall wäre das Einbringen von neuen Komponenten in den Kommunikationsweg zwischen Sensoren/Aktoren und Softwarekomponenten, um eben diesen Weg über Switches oder Repeater detaillierter zu spezifizieren.

Wenn die Komponente andererseits atomar ist, also keine weiteren Unterkomponenten besitzt, muss weiterhin die Frage beantwortet werden, ob die vorliegende Verfeinerungsaufgabe so komplex ist, dass eine weitere Zerlegung notwendig ist. In diesem Fall muss die Komponente weiter aufgebrochen werden, wodurch die ursprünglich atomare Komponente zu einer zusammengesetzten Komponente wird. Die Verhaltensbeschreibung dieser Komponente kann dabei umfunktioniert werden und fortan als Monitor für die detailliertere Verhaltensbeschreibung mittels der neu generierten Unterkomponenten dienen. Ein Beispiel für diesen Fall ist ein Sensor, der zunächst noch sehr abstrakt über einen Detektor-Port modelliert wurde, während tatsächlich ein kamerabasiertes System mit entsprechenden Bildverarbeitungsalgorithmen zum Einsatz kommen soll. Der Sensor wird in diesem Fall in eine Kamera- und eine Softwarekomponente zerlegt.

Wenn andererseits keine Unterkomponenten für die Verfeinerung der Komponente benötigt werden, muss stattdessen der Inhalt der Komponente durch eine detaillierte Spezifikation ersetzt werden. Dafür kann eine Modelltransformation des Komponenteninhalts durchgeführt werden, der dann als Basis für die Komponentenausarbeitung verwendet werden kann. Dieser Fall kann bspw. auftreten, wenn der Zustandsautomat einer Softwarekomponente das gewünschte Verhalten nur teilweise erfasst. In diesem Fall kann der Automat in IEC 61131-3 Code transformiert werden, der dann in einer SPS-Entwicklungsumgebung verfeinert werden kann.

Sobald die betrachtete Komponente vollständig ausgearbeitet wurde, kann eine Co-Simulation durchgeführt werden, um die syntaktische und die semantische Korrektheit des mechatronischen Gesamtsystems zu überprüfen. Im Speziellen kann mittels der bereits aus der Konzeption bekannten Qualitätssicherungsmaßnahmen gezeigt werden, ob neue Komponenten korrekt eingebettet wurden (syntaktische Korrektheit) und ob das verfeinerte Systemmodell nach wie vor den spezifizierten Szenarien, Monitoren und Bedingungen genügt (semantische Korrektheit). Wenn einer dieser Aspekte nicht eingehalten ist, schlägt der erarbeitete Workflow ein iteratives Vorgehen vor, beginnend mit der Überarbeitung der Spezifikationsanpassung. Wenn die Qualitätssicherung andererseits die Korrektheit der Implementierung bestätigt, ist die Verfeinerung für diese Komponente abgeschlossen und das inkrementelle Vorgehen kann mit einer erneuten Analyse des mechatronischen Konzeptes fortgesetzt werden. Sobald keine weiteren Komponenten für die Verfeinerung mehr gefunden werden, ist auch dieser Entwicklungsschritt abgeschlossen.

4 Evaluation

Im vierten Kapitel des Abschlussberichtes werden die einzelnen Aspekte der modellbasierten Entwicklungsmethodik an dem Fallbeispiel des Bedruckungsmoduls einer miniaturisierten Verpackungsanlage evaluiert. Dabei wird zunächst in Abschnitt 4.1 das gewählte Beispiel vorgestellt sowie die Vorgehensweise beschrieben, wie die Evaluation unter Einbeziehung des projektbegleitenden Ausschusses durchgeführt wurde. Darauf aufbauend wird in Abschnitt 4.2 dargelegt, wie das Fallbeispiel unter Verwendung der modellbasierten Entwicklungsmethodik realisiert wurde. Die Ergebnisse werden schließlich in Abschnitt 4.3 kritisch reflektiert und bewertet.

4.1 Fallbeispiel und Vorgehensweise für die Evaluation

Zur Evaluierung und Bewertung von unterschiedlichen mechatronischen Entwicklungsprinzipien wird am Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) der Technischen Universität München sowie an der Fraunhofer-IWU-Projektgruppe für Ressourceneffiziente Mechatronische Verarbeitungsmaschinen derzeit eine miniaturisierte Verpackungsanlage aufgebaut. Die Funktion dieser Anlage liegt darin, zunächst aufgerolltes Papier zu Schachteln zu formen, die anschließend mit einem speziellen Stückgut befüllt werden. Zur Erfüllung dieser Funktion gliedert sich die Anlage in einzelne, weitestgehend autark agierende Module, die die einzelnen Prozessschritte zur Herstellung und zum Befüllen der Verpackung realisieren. Eines dieser Module übernimmt dabei die Aufgabe, ein zu diesem Zeitpunkt bereits ausgestanztes Papier mit den Logos der oben genannten Institutionen zu bedrucken und den bedruckten Papierrohling anschließend an das nachfolgende Modul weiterzuleiten (vgl. Abbildung 46).

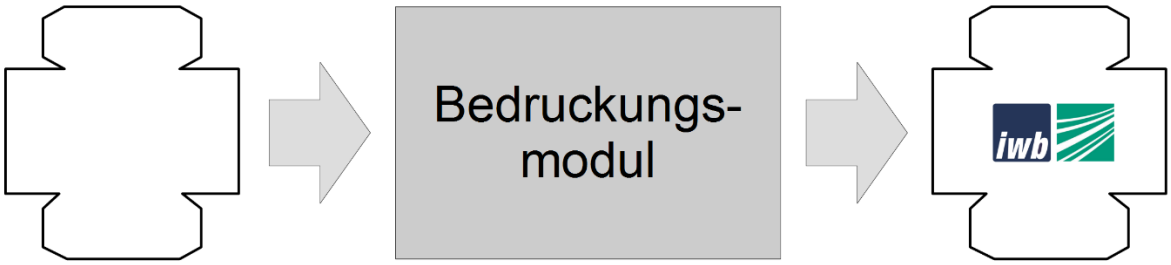


Abbildung 46: Funktion des Bedruckungsmoduls

Bei der Entwicklung dieses Bedruckungsmoduls der miniaturisierten Verpackungsanlage wurde die in Kapitel 3 vorgestellte modellbasierte Entwicklungsmethodik angewandt. Während des gesamten Entwicklungsprozesses wurden die Zwischenergebnisse der Modellierung dem projektbegleitenden Ausschuss kontinuierlich in den Statustreffen vorgestellt sowie als weitere Evaluationsmaßnahme im Rahmen einzelner Workshops mit Entwicklern aus den Bereichen Mechanik, Elektrik und Steuerungstechnik diskutiert. Das grundsätzliche Ziel war zunächst die Bewertung der einzelnen Inhalte des IMoMesA-Ansatzes hinsichtlich ihrer Bedeutung für einen realen Entwicklungsprozess im Werkzeugmaschinenbau und eine diesbezügliche Einteilung in Stärken und Schwächen. Außerdem konnten mögliche Erweiterungen des Ansatzes identifiziert werden, die in die Modellierungstechnik integriert werden müssen, um eine langfristige Etablierung eines solchen Entwicklungsansatzes im Werkzeugmaschinenbau sicherzustellen. Diese verschiedenen Aspekte der Evaluation werden in Abschnitt 4.3.1 jeweils für die Konzeption und die Verfeinerung vorgestellt.

4.2 Anwendung der modellbasierten Entwicklungsmethodik

Im folgenden Abschnitt werden die einzelnen Aspekte des IMoMeSA-Ansatzes bei der Konzeption und Realisierung des eben eingeführten Bedruckungsmoduls beleuchtet. Analog zum Aufbau der Modellierungstechnik werden die Arbeiten am Fallbeispiel grundsätzlich in die Phasen der Konzeption und Verfeinerung unterteilt. Es sei an dieser Stelle darauf hingewiesen, dass es nicht das Ziel der folgenden Abschnitte ist, die vollständige Modellierung des Fallbeispiels darzulegen, da einzelne Inhalte des IMoMeSA-Ansatzes oft mehrfach angewandt wurden, was für das Verständnis und die Bewertung keinen Mehrwert bietet. Vielmehr sind die folgenden Abschnitte so strukturiert, dass die Anwendung der Inhalte der Entwicklungsmethodik exemplarisch aufgezeigt wird, wobei der Bezug zum Fallbeispiel jedoch stets ersichtlich bleibt.

4.2.1 Konzeption des Bedruckungsmoduls

In Anlehnung an den definierten Workflow bei der Konzeption mechatronischer Systeme (vgl. Abschnitt 3.1.3) wird nachfolgend zunächst die Modellierung des Bedruckungsmoduls hinsichtlich seines Normalverhaltens vorgestellt (Abschnitt 4.2.1.1). Darauf aufbauend wird dargelegt, inwieweit die einzelnen Maßnahmen zur Qualitätssicherung in dieser Phase der Entwicklung Anwendung fanden (Abschnitt 4.2.1.2), bevor schließlich exemplarisch ein in das System eingebrachter und behandelter Komponentenfehler beschrieben wird (Abschnitt 4.2.1.3).

4.2.1.1 Modellierung des Bedruckungsmoduls

Die Entwicklung des Bedruckungsmoduls mittels des IMoMeSA-Ansatzes begann mit der Definition textueller Anforderungen. Für das gewählte Fallbeispiel beinhaltete dies Anforderungen an die Zykluszeit, den maximalen Energieverbrauch sowie die Übergabepositionen der Papierrohlinge. Die letztgenannten Anforderungen konnten anschließend direkt verwendet werden, um die Schnittstelle des Bedruckungsmoduls zu seiner Umgebung zu modellieren. Dabei wurden mittels zweier Detector-Ports Positionen im dreidimensionalen Raum (Start- und Endposition, vgl. Abbildung 47a) sowie ein elektrischer Energie-Port „Energieversorgung“ spezifiziert. Auf dieser Basis konnte ein erstes Szenario abgeleitet werden, welches die Interaktion des Moduls mit seiner Umgebung beschreibt. Dafür definiert die Umgebung einen neuen Material-eingangs-Port, der notwendig ist, um während der Ausführung des Szenarios ein Papier zu erzeugen. Anschließend wird darauf gewartet, dass ein bearbeitetes Papier vom Bedruckungsmodul an die Umgebung zurückgegeben wird (vgl. Abbildung 47b). Darüber hinaus konnte bereits eine erste Bedingung modelliert werden, die die Anforderungen an die Zykluszeit formalisiert und dadurch prüfbar macht.

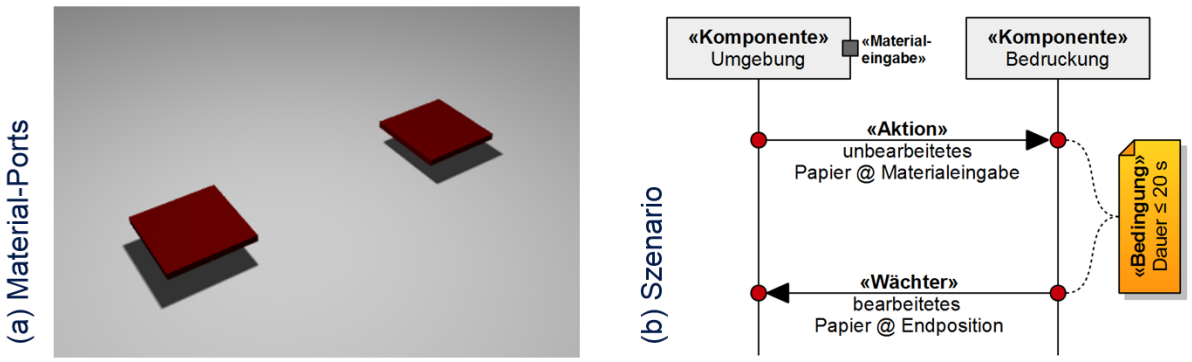


Abbildung 47: Material-Ports und Szenario des Bedruckungsmoduls

Auf Basis des Szenarios konnte im nächsten Schritt ein Monitor abgeleitet werden, der den Prozess des Bedruckungsmoduls grundsätzlich in eine Aktivität „Warten“ und eine Aktivität „Bearbeitung“ unterteilt. Das Bedruckungsmodul wechselt dabei die Aktivität von „Warten“ zu „Bearbeitung“, sobald ein Papierrohling an der Start-Position erkannt wird und gleichzeitig kein Material mehr an der End-Position anliegt. Andererseits erfolgt der Wechsel zurück in die „Warten“-Aktivität, sobald ein bearbeitetes Papier die Endposition erreicht. Neben der Verknüpfung der bereits im Szenario vorhandenen zeitlichen Bedingung mit der Aktivität „Bearbeitung“ wurde ebenfalls eine Qualität bezüglich des Energieverbrauchs des Bedruckungsmoduls modelliert, die kontinuierlich den zuvor definierten Energie-Port beobachtet (vgl. Abbildung 48).

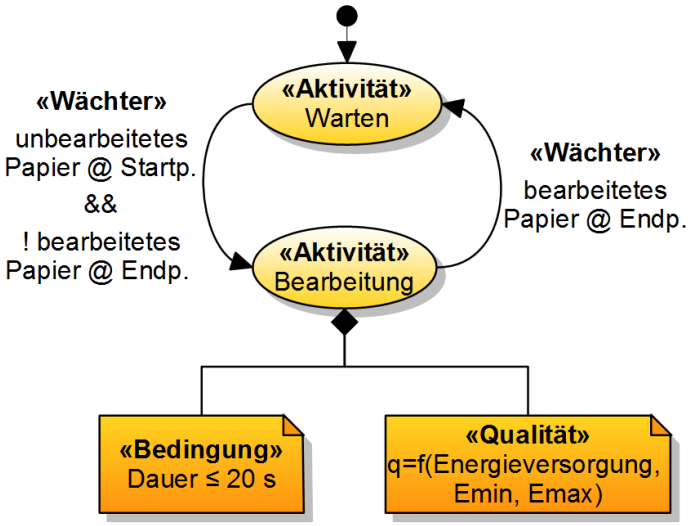


Abbildung 48: Monitor des Bedruckungsmoduls

An diesem Punkt der Entwicklung wurde entschieden, dass der zu realisierende Prozess eine zu große Komplexität aufweist, um ihn direkt zu implementieren. Da der Monitor offenlegt, dass grundsätzlich zwei Hauptaufgaben erfüllt werden müssen (Überführung des Papiers von der Start- zur Endposition und vom Zustand „unbearbeitet“ in den Zustand „bearbeitet“), wurde eine Zerlegung des Bedruckungsmoduls in die beiden Unterkomponenten „Transport“ und „Bearbeitung“ vorgenommen. Abbildung 49 visualisiert diese Zerlegung und verdeutlicht gleichzeitig die Modellierung der Schnittstelle dieser Unterkomponenten sowie deren Verknüpfung durch Kanäle. Die Transport-Komponente besitzt zunächst analog zum Bedruckungsmodul zwei Detector-Ports (Start- und Endposition). Da sich bei Material-Ports eine Verbindung implizit durch eine geometrische Überschneidung ergibt, mussten an dieser Stelle keine eigenen Kanäle für die Verbindung mit den Ports der übergeordneten Komponente modelliert werden. Neben diesen beiden Material-Ports wurde über einen dritten Detector-Port eine neue Mittel-Position definiert, an der die Transport-Komponente das Material an die Bearbeitungs-Komponente übergibt. Auch hier erfolgt die Verknüpfung implizit aus einer geometrischen Überschneidung. Darüber hinaus besitzen beide Subkomponenten eigene Energie-Ports, um ihren jeweiligen Energieverbrauch an die Energieversorgung zu melden. Die Verbindung zwischen diesen Ports wird in diesem Fall explizit über zwei Kanäle („Energiebedarf“) modelliert. Darüber hinaus besitzt jede der Komponenten schließlich zwei Daten-Ports, über welche die Informationen ausgetauscht werden können, ob ein Material zugeführt

oder ob ein Bearbeitungsvorgang abgeschlossen wurde. Wie bei den Energie-Ports wurde die Verbindung dieser Ports explizit mittels zweier Kanäle („zugeführt“ und „bearbeitet“) bestimmt.

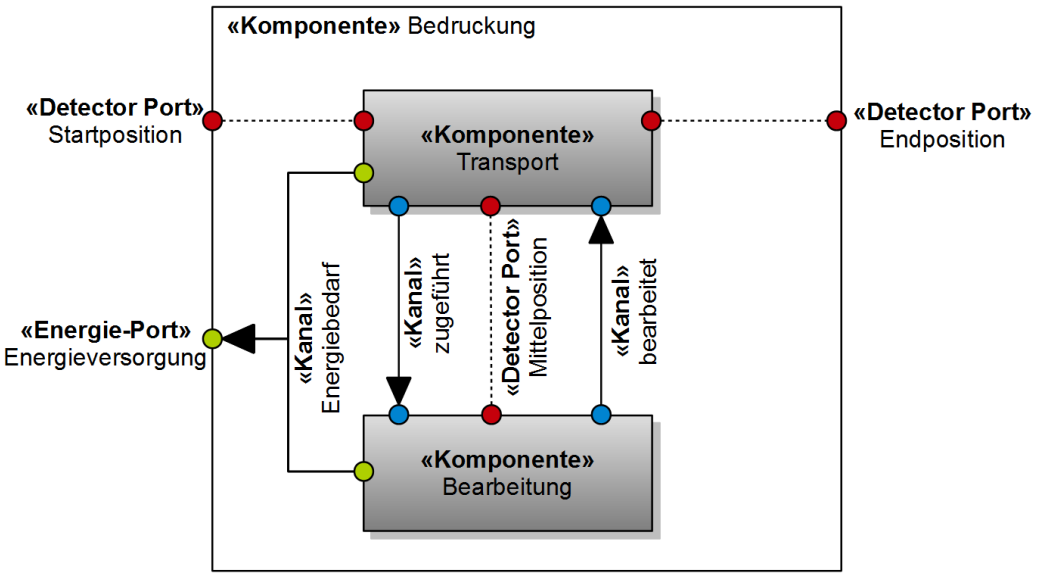


Abbildung 49: Zerlegung des Bedruckungsmoduls

Gemäß dem rekursiven Entwicklungsworkflow wurden anschließend die beiden abgeleiteten Komponenten ausgearbeitet. Aufgrund des komponentenbasierten Ansatzes mit einer klaren Kapselung der Funktionalität konnten diese Komponenten weitestgehend losgelöst voneinander entwickelt werden. Da die Modellierung bei beiden Unterkomponenten sehr ähnlich ablief, wird im Folgenden nur die weitere Ausarbeitung der Transport-Komponente vorgestellt.

Für diese Komponente wurden zunächst wiederum informelle Anforderungen gesammelt und modelliert. Diese konnten teilweise aus den übergeordneten Anforderungen an das Bedruckungsmodul abgeleitet werden, aber auch neue Anforderungen, wie die Festlegung der Übergabeposition, wurden definiert. Auf Basis der bereits zuvor erklärten Schnittstelle konnten anschließend ein Szenario und ein Monitor zur Spezifikation der geforderten Interaktion sowie der Prozessschritte modelliert werden. Für das Verständnis der Funktionalität der Transportkomponente genügt es allerdings, im Folgenden alleine den modellierten Monitor näher zu betrachten (vgl. Abbildung 50). Auch die Transport-Komponente beginnt zunächst mit der Aktivität „Warten“. Sobald ein Material an der Start-Position anliegt, wird in die Aktivität „Zuführen“ gewechselt. Diese wiederum bleibt solange aktiv, bis ein Papier an der Mittel-Position erkannt und das Signal „zugeführt“ gesendet wird. Daraufhin wartet die Transport-Komponente solange, bis sie den Befehl zum Abführen des Materials erhält, welches ihr über den Kanal „bearbeitet“ des Bedruckungsmoduls kommuniziert wird. Sobald dies der Fall ist, wechselt die Komponente in den Zustand „Abführen“, den sie schlussendlich wieder verlässt, wenn ein Papier an der End-Position erkannt wurde. Neben der reinen Prozessbeschreibung konnten in dieser Phase ebenfalls wiederum Bedingungen definiert und mit einzelnen Aktivitäten verknüpft werden. Abbildung 50 zeigt exemplarisch je eine Bedingung an die Aktivitäten „Zuführen“ und „Abführen“, für die jeweils eine maximale Verweildauer vorgegeben wird.

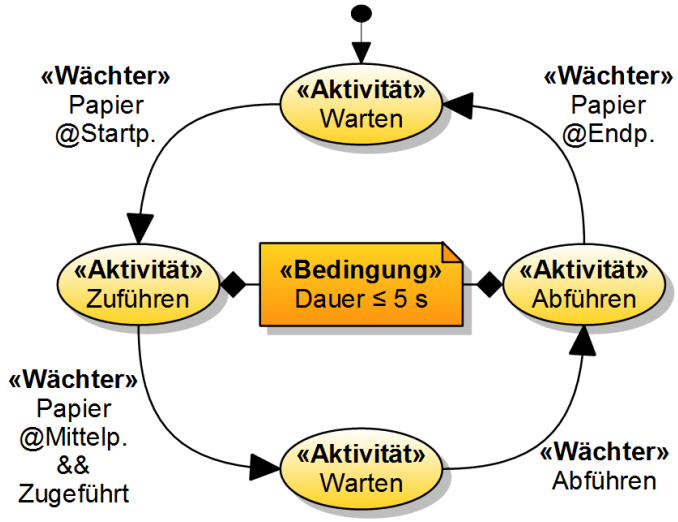


Abbildung 50: Monitor der Transport-Komponente

Auch bei der Transportkomponente wurde auf Basis des modellierten Monitors entschieden, die Komponente zur Verringerung der Komplexität weiter zu zerlegen. Abbildung 51 zeigt die resultierende Komponentenstruktur, bestehend aus zwei Lichtschranken an der Start- und Endposition, einem Förderband mit zugehörigem Antrieb, einem Material-Stopper mit zugehörigem Ventil sowie einer Steuerungskomponente mit eigenem Timer. Dem Material-Stopper kommt dabei die Aufgabe zu, das Material an der bereits zuvor definierten Mittelposition auszurichten, um so eine genaue Bearbeitung des Papiers zu ermöglichen. Darüber hinaus kamen durch die Zerlegung des Transportmoduls zwei weitere Binding-Ports für das Förderband und den Material-Stopper hinzu, die auch an der Schnittstelle des Transport-Moduls anliegen und dafür verantwortlich sind, kollidierendes Papier an das Förderband bzw. den Material-Stopper zu binden. Über entsprechende kinematische Energie-Ports wirkt auf diese Material-Ports ein bestimmter Energiewert, der angibt, mit welcher Geschwindigkeit sich kollidierendes Material auf den jeweiligen Komponenten bewegt. Zur Verbindung der einzelnen Unterkomponenten kamen zudem eine Vielzahl an neuen Energie- und Daten-Ports sowie zugehöriger Kanäle zum Einsatz, deren Bedeutungen direkt Abbildung 51 entnommen werden können.

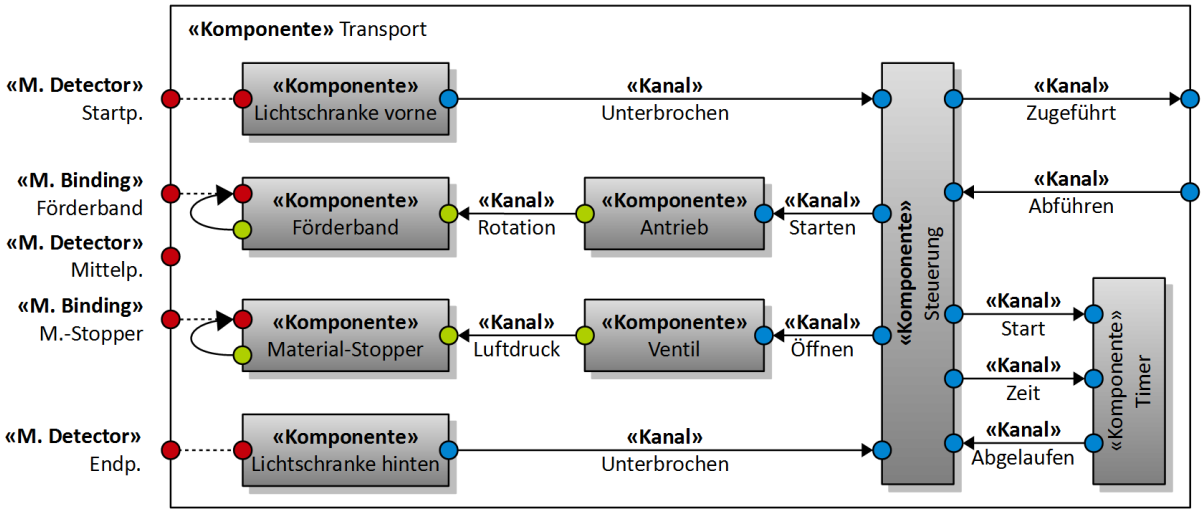
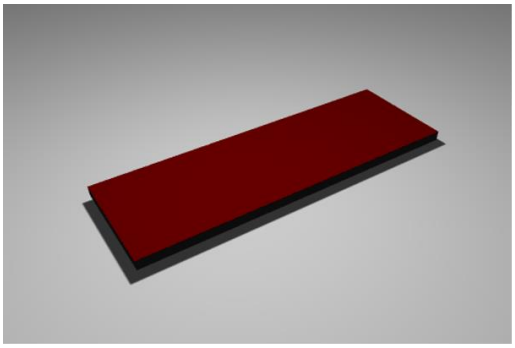


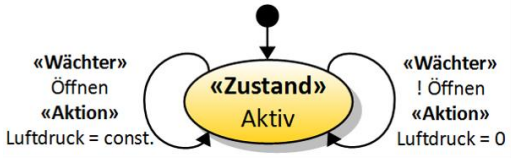
Abbildung 51: Zerlegung der Transport-Komponente

Aus Platzgründen wurde in der Abbildung ein Kompressor, der die Druckluft für das Ventil bereitstellt, weggelassen. Auch die Modellierung des Energieverbrauchs ist nicht mehr dargestellt. Dieser wurde analog zum Bedruckungsmodul so modelliert, dass die energieverbrauchenden Komponenten (Kompressor und Antrieb) ihre jeweiligen Energiebedarfe an den Energie-Port der Transport-Komponente melden.

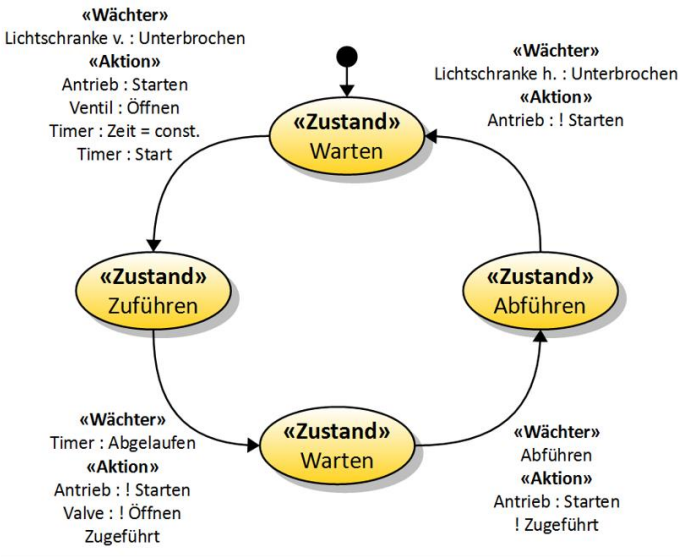
Für jede der durch den vorigen Schritt abgeleiteten Komponenten konnten anschließend die einzelnen Modellierungsschritte aus der Analyse- und Designphase wiederholt werden, um zu ermitteln, ob jeweils eine weitere Zerlegung notwendig ist. Dabei zeigte sich, dass alle in Abbildung 51 dargestellten Komponenten atomar sind und somit ohne weitere Zerlegung implementiert werden konnten. Wie in Kapitel 3 deutlich wurde, setzt sich die Implementierungsphase schließlich aus der Modellierung von Verhalten und Bauteilen zusammen. Nachfolgend werden exemplarisch die jeweiligen Verhaltensimplementierungen des Ventils und der Steuerungs-Komponente sowie die Bauteilmodellierung für das Förderband vorgestellt (vgl. Abbildung 52).



(a) Förderband mit Binding-Port



(b) Verhalten des Ventils



(c) Verhalten der Steuerung

Abbildung 52: Implementierungen ausgewählter Unterkomponenten

Da die Modellierung von Bauteilen bewusst auf Grob-Geometrien eingeschränkt wurde, wird das Förderband in diesem Fall vereinfacht durch einen Quader modelliert, der lediglich die ungefähren Maße des Förderbands definiert (vgl. Abbildung 52a). Darüber hinaus ist auf der Oberfläche des Quaders der Material-Binding-Port abgebildet, der in diesem Fall die gleiche Länge und Breite wie das Förderband selbst besitzt. Das Verhalten des Förderbands ist schließlich über einen Zustandsautomaten definiert (nicht Teil der Abbildung), der mittels seines kinematischen Energie-Ports translatorische Energie an den Binding-Port überträgt, sofern eine entsprechende Energie vom Antrieb am Förderband anliegt.

Abbildung 52b zeigt stattdessen die Modellierung des Ventils. Dabei reichen ein einzelner „Aktiv“-Zustand sowie zwei Transitionen aus, um das Verhalten auf Konzeptebene zu beschreiben: Sobald das Datensignal „Öffnen“ beobachtet werden kann, wird ein konstanter Wert an den entsprechenden Energie-Port angelegt. Wenn andererseits das Datensignal nicht anliegt, wird der Wert des Luftdrucks auf null

gesetzt. Schließlich zeigt Abbildung 52c das Verhalten der modellierten Steuerung. Diese besitzt grundsätzlich den gleichen Aufbau wie der Monitor der Transportkomponente. Die Unterschiede ergeben sich im Wesentlichen durch die zusätzlichen Datensignale, die mit den übrigen Komponenten auf dieser Hierarchieebene ausgetauscht werden. Unter diesen ist insbesondere der Timer hervorzuheben, der dafür verantwortlich ist, das Förderband beim Zuführen nach einer definierten Zeit wieder anzuhalten, da für diese Materialposition kein eigener Sensor vorgesehen wurde.

Auf vergleichbare Weise konnten die Lichtschranken, der Antrieb, der Material-Stopper und der Timer sowie die einzelnen Unterkomponenten der hier nicht weiter betrachteten Bearbeitungs-Komponente implementiert werden. Dadurch konnten mittels eines Top-Down-Vorgehens die atomaren Bestandteile des zu entwickelnden Bedruckungsmoduls auf Konzeptebene spezifiziert werden. Im letzten Schritt des definierten Vorgehens mussten die ausgearbeiteten Komponenten schließlich mittels eines Bottom-Up-Vorgehens zu einem konsistenten Gesamtsystem integriert werden. Die Integration auf der Ebene der atomaren Komponenten beschränkte sich zunächst auf die Verknüpfung der einzelnen Ports mittels unterschiedlicher Kanäle (wie bereits exemplarisch in Abbildung 51 dargestellt). Zur Integration der Transport- und Bearbeitungskomponente hingegen mussten eine gemeinsame Bodenplatte für die jeweiligen Bauteile eingebracht sowie eine Gesamtsteuerung modelliert werden, die für die Aktivierung der einzelnen Sensoren und Aktoren in Abhängigkeit des gesamten Anlagenstatus benötigt wurde. Abbildung 53 gibt abschließend einen Überblick über die gesamten modellierten Bauteile (inkl. der Bodenplatte) sowie über die Verhaltens-implementierung der Gesamtsteuerung.

Abbildung 53: Gesamtgeometrie und Gesamtsteuerung des Bedruckungsmoduls

4.2.1.2 Beispielhafte Anwendung der Qualitätssicherungsmaßnahmen

Nachfolgend wird beschrieben, wie die einzelnen Maßnahmen zur Qualitätssicherung (i.e. Kompilierung, Simulation, Unit-Testen, kontinuierliche Integration) die Entwickler bei der Konzeption des Bedruckungsmoduls unterstützten. Auch in diesem Abschnitt wird die Anwendung der einzelnen Maßnahmen nur exemplarisch an ausgewählten Beispielen dargelegt, die zum Verständnis bestmöglich beitragen können.

Kompilierung

In einer frühen Phase der Konzeption wurden für die einzelnen Antriebe des Bedruckungsmoduls komplexe Servo-Motoren modelliert, die zur Steuerung einen analogen Soll-Geschwindigkeitswert benötigen. In einer späteren Phase der Entwicklung entschieden sich die Entwickler der Elektrik aus Kostengründen dazu,

stattdessen einen einfacheren Motor zu verwenden, der zur Steuerung lediglich einen Booleschen Eingabewert „Start“ benötigt. Durch eine Kompilierung des Gesamtsystems im Anschluss an den Austausch der modellierten Antriebe wurde dabei offensichtlich, dass der neue Antrieb nicht mehr kompatibel zu der bereits entworfenen Steuerungskomponente war. Durch die frühzeitige Identifikation dieser Inkompatibilität konnte bereits zu diesem Zeitpunkt die Steuerungskomponente an die neuen Begebenheiten angepasst werden und der Fehler wurde nicht (wie aktuell häufig der Fall) erst bei der Inbetriebnahme des Gesamtsystems erkannt.

Simulation

Bei der gegebenen Modellierung des Bedruckungsmoduls konnte die Simulation hauptsächlich eingesetzt werden, um das grundsätzliche Zusammenspiel der einzelnen Komponenten im Verbund zu visualisieren sowie um zu überprüfen, ob das System den Bearbeitungsvorgang in der gewünschten Zeit und mit einem tolerablen Energieverbrauch durchführen kann. Ein exemplarisches Beispiel, bei dem durch die Simulation ein Implementierungsfehler offengelegt werden konnte, bezieht sich erneut auf den modellierten Antrieb: Durch den Wechsel des Antriebs zu den einfacheren Antrieben musste neben der Änderung an den Daten-Ports und Kanälen auch das Verhalten des Antriebs angepasst werden, so dass nur noch weniger kinematische Energie vom Antrieb auf das Förderband übertragen werden konnte. In der Simulation zeigte sich, dass diese Änderung dazu führte, dass der Timer das Förderband anhielt, bevor das Material durch den Material-Stopper an der Mittelposition ausgerichtet werden konnte. Durch eine Verlängerung der Aktivierungszeit des Timers von drei auf vier Sekunden konnte dieser Implementierungsfehler behoben werden. Abbildung 54 zeigt exemplarisch zwei Screenshots aus dem „IMoMeSA Analyzer“, auf denen ein typischer Simulationsablauf für eine frühe Variante des Bedruckungsmoduls bzw. des dabei auftretenden Energieverbrauchs visualisiert ist.

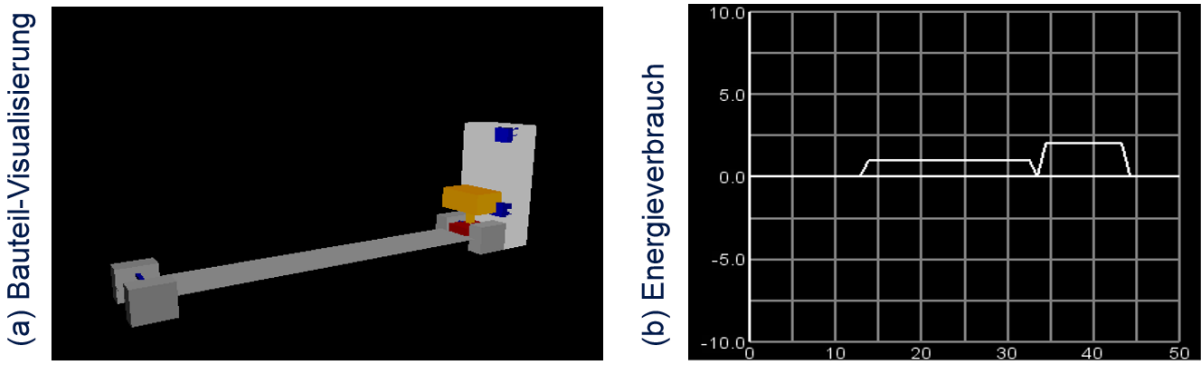


Abbildung 54: Visualisierung der Simulation des Bedruckungsmoduls

Unit-Testen

Bei der Konzeption des Bedruckungsmoduls wurden insgesamt sechs Szenarien spezifiziert. Neben dem in Abbildung 47c dargestellten Szenario für das Gesamtsystem wurden zwei weitere Szenarien für den vom Rest des Systems isolierten Test der Transport- und Bearbeitungskomponente spezifiziert sowie drei weitere Szenarien für die als am kritischsten eingestuften atomaren Komponenten Förderband, Material-Stopper und Stempel. Da die Simulation einer konkreten Komponente nur dann möglich ist, wenn diese oder (im Falle einer weiteren Zerlegung) ihre Unterkomponenten bereits implementiert sind, war das Unit-Testen am Ende des Top-

Down-Vorgehens zunächst nur für die drei atomaren Komponenten möglich. Nach Absicherung der Funktionsweise dieser Komponenten konnten die einzelnen Unterkomponenten integriert werden, wodurch das Unit-Testen auch auf die beiden Szenarien der Transport- und Bearbeitungskomponente ausgeweitet werden konnte. Nach Integration dieser Komponenten mittels der eingeführten Bodenplatte und der Gesamtsteuerung konnte schlussendlich auch überprüft werden, ob das modellierte System auch den eingangs formulierten Kundenanforderungen gerecht wird.

Kontinuierliche Integration

Schließlich konnte auch der Mechanismus der kontinuierlichen Integration dazu beitragen, den Fortschritt über die Projektlaufzeit zu dokumentieren. Nach der Zerlegung der Transport- bzw. Bearbeitungskomponente gab es zunächst eine Reihe von atomaren Komponenten, die noch nicht implementiert waren und somit auch nicht für eine Simulation zur Verfügung standen. Schritt für Schritt konnten die Entwickler für einzelne Komponenten ein Verhalten bzw. Bauteile modellieren, bis schließlich alle atomaren Komponenten implementiert wurden. Durch die kontinuierliche Integration konnte die Anzahl der noch nicht implementierten Komponenten in Form von kurzen Management-Reports an die Projektleitung gespiegelt werden, die dadurch auf einfache Weise kontinuierlich den Projektfortschritt überwachen konnte.

4.2.1.3 Modellierung und Behandlung eines Komponentenfehlers

Nach der Absicherung des Normalverhaltens des Bedruckungsmoduls mittels der eben skizzierten Qualitätssicherungsmaßnahmen konnte das erarbeitete Konzept durch die Modellierung und Behandlung von Komponentenfehlern in seiner Robustheit gesteigert werden. Nachfolgend wird dieser Vorgang exemplarisch für die Ventil-Komponente beschrieben.

Gemäß dem vorgeschlagenen Workflow (vgl. Abschnitt 3.1.3.2) wurde in einem ersten Schritt der Fehlermodellierung und -behandlung ein alternativer Verhaltensmodus für das Ventil definiert. Für diesen Modus konnte anschließend ein Zustandsautomat modelliert werden, der das Verhalten der Komponente im Fehlerfall festlegt (vgl. Abbildung 55a). Im Unterschied zum Normalverhalten wird im Fehlerfall unabhängig vom Wert des anliegenden Datensignals „Öffnen“ der Energie-Port „Luftdruck“ auf den Wert null gesetzt. Zum Wechsel zwischen den beiden Modi (Normal- und Fehlerfall) mussten der Ventil-Komponente zudem zwei Ereignis-Ports („defekt“ und „repariert“) sowie zwei Transitionen hinzugefügt werden. Die Ereignis-Ports dienen dabei als Wächter zum Schalten der Transitionen und ermöglichen somit einen von der Umgebung veranlassten Wechsel zwischen den beiden Verhaltensmodi. Um den alternativen Verhaltensmodus schließlich aktivieren zu können, muss der entsprechende Ereignis-Port in einem bestehenden Szenario getriggert werden. Im Fallbeispiel wurde dazu das in Abbildung 47b bereits vorgestellte Szenario des Bedruckungsmoduls verwendet, in dem unmittelbar nach dem Einlegen eines Papiers an den Material-Eingabe-Port der Ereignis-Port „defekt“ aktiviert wird (vgl. Abbildung 55b). Ansonsten blieb das modellierte Szenario zunächst unverändert. Die ursprünglich modellierte Bedingung an die Dauer des Bearbeitungsvorgangs ist in Abbildung 55b lediglich aus Übersichtsgründen nicht dargestellt.

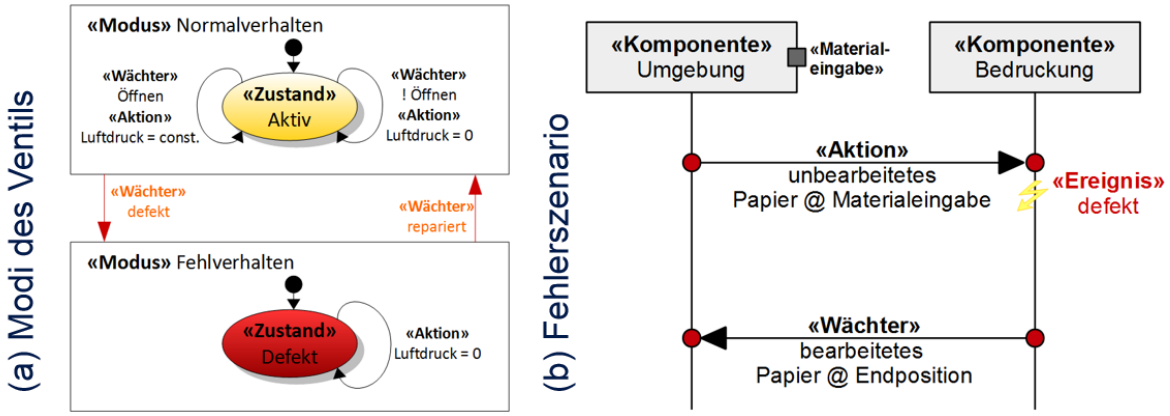


Abbildung 55: Modi und Fehlerszenario der Ventil-Komponente

In der Simulation konnte auf Basis der Fehlermodellierung geprüft werden, welche Auswirkungen der eingebrachte Fehler auf das gesamte Systemverhalten hat. Dadurch, dass kein Luftdruck an den Material-Stopper übergeben wird, konnte dieser nicht ausfahren, was schließlich dazu führte, dass das Papier nicht in der definierten Mittelposition zum Stehen kam und somit nicht bearbeitet werden konnte. Somit konnte das getestete Szenario nicht mehr den formulierten Anforderungen gerecht werden, da der Bearbeitungsvorgang nicht in der geforderten Zeit beendet werden konnte. Aus diesem Grund musste in einem zweiten Schritt eine entsprechende Fehlerbehandlung in das modellierte System eingebracht werden.

Gemäß dem in Abbildung 31 dargestellten Vorgehen muss zunächst sichergestellt werden, dass ein eingebrachter Komponentenfehler während des späteren Systembetriebs auch erkannt wird. Durch eine Fehlerdiagnose (vgl. Abschnitt 3.1.4) konnte überprüft werden, ob der eingebrachte Fehler auf Basis zur Verfügung stehender Datensignale erkannt werden kann. Da dies im Fallbeispiel nicht möglich war, musste ein Endlagensensor in das System eingebracht werden, dessen Aufgabe es ist, über einen Material-Port zu detektieren, ob der Material-Stopper während des Zuführvorgangs ausgefahren werden konnte (vgl. Abbildung 56).

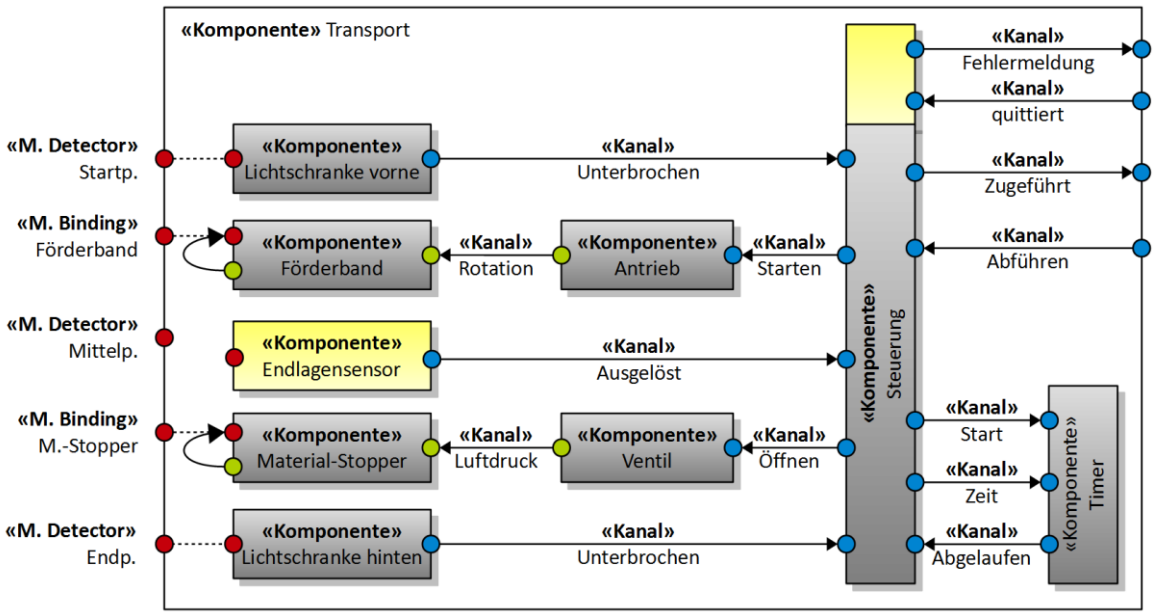


Abbildung 56: Erweiterung der Transport-Komponente um einen Endlagensensor

Das von diesem Sensor zur Verfügung gestellte Datensignal konnte schließlich in der Steuerungskomponente berücksichtigt werden, um eine entsprechende Fehlerbehandlung zu implementieren. Dabei wurden in den Zustandsautomat der Steuerung ein neuer Zustand sowie eine neue Transition eingebracht, die überprüft, ob nach dem Ablauf des Timers der Endlagensensor ausgelöst hat. Ist dies nicht der Fall, wird das Transportband angehalten und eine Fehlermeldung ausgegeben (vgl. Abbildung 57). Um schließlich in den Gutablauf zurückzukehren, bedarf es der Interaktion eines Werkers mit der Komponente, der diese repariert oder austauscht. Über eine „Quittiert“-Meldung kann schließlich in den Ausgangszustand zurückgekehrt werden.

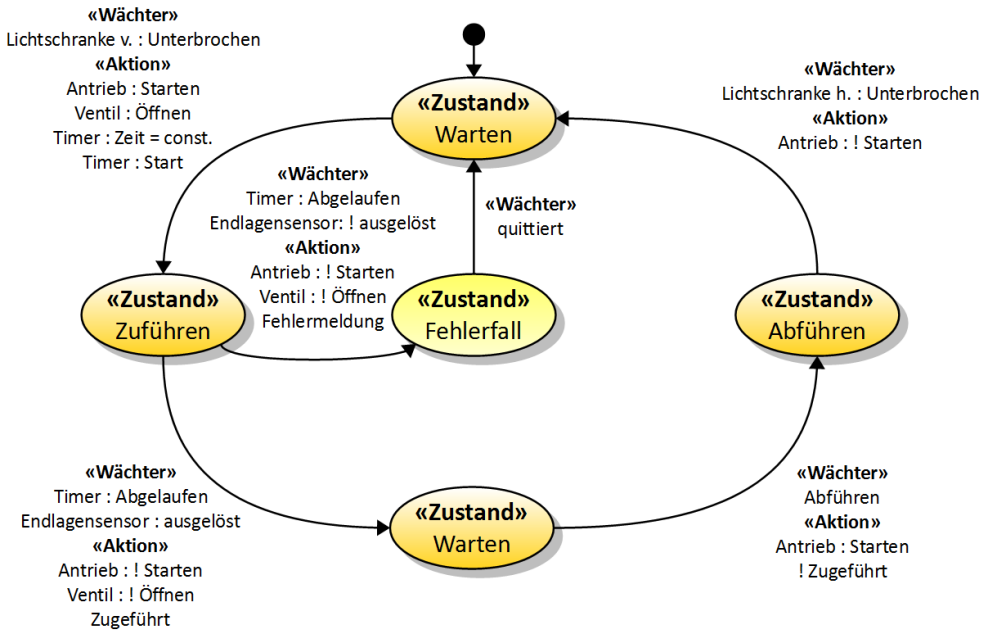


Abbildung 57: Erweiterung der Steuerungskomponente

In einer erneuten Simulation des Gesamtsystems verhielt sich das System nun zwar korrekt, allerdings wurde dieselbe Bedingung verletzt wie bei der ersten Simulation während der Fehlermodellierung. Aus diesem Grund mussten im letzten Schritt des in Abschnitt 3.1.3.2 vorgestellten Workflows die Spezifikation des Bedruckungsmoduls sowie der Transport-Komponente angepasst werden. Im Detail musste zunächst das Fehlerszenario angepasst werden (vgl. Abbildung 58), in dem nun anstelle eines bearbeiteten Materials am Ende eine Fehlermeldung innerhalb einer vorgeschriebenen Zeit an die Umgebung ausgegeben werden muss. Über ein anschließendes Ereignis „repariert“ verbunden mit dem Datensignal „quittiert“ kann das System schließlich in seinen Ausgangszustand zurückgesetzt werden. Neben dem Szenario mussten auch die Monitore des Bedruckungsmoduls sowie der Transport-Komponente angepasst werden, indem analog zur Steuerungskomponente jeweils eigene Fehlerzustände mit entsprechenden Übergangsbedingungen hinzugefügt wurden. In einer abschließenden Simulation wurden trotz des Auftretens des modellierten Komponentenfehlers keine Bedingungen mehr verletzt, so dass die Fehlerbehandlung für diesen exemplarischen Verhaltensfehler beendet war.

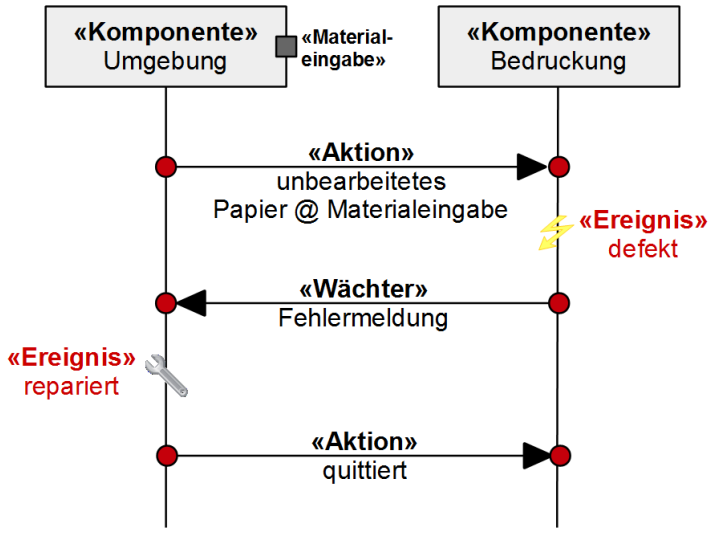


Abbildung 58: Angepasstes Fehlerszenario des Bedruckungsmoduls

4.2.2 Verfeinerung des Bedruckungsmoduls

Nach der Entwicklung eines robusten mechatronischen Konzepts konnten in der Phase der Verfeinerung diese Erkenntnisse genutzt werden, um einen vollständigen virtuellen Prototyp des Bedruckungsmoduls auszuarbeiten. Da der Co-Simulationsansatz über das Functional Mockup Interface [49] im Rahmen des Forschungsprojektes lediglich konzipiert, aber nicht realisiert wurde, konnten die Qualitätssicherungsmaßnahmen des „IMoMeSA Analyzer“ im Rahmen der Verfeinerung nicht mehr genutzt werden. Aus diesem Grund beschränkten sich die Arbeiten am Fallbeispiel während der Verfeinerung auf die Ausarbeitung der im IMoMeSA-Ansatz definierten Deployment Units. Vor diesem Hintergrund wird in den folgenden Abschnitten die Entwicklung der MCAD- und ECAD-Modelle sowie des Steuerungsprojektes auf Basis der vorgestellten Modelltransformationen und der definierten Vorgehensweise vorgestellt. Auch in diesen Teilabschnitten beschränken sich die nachfolgenden Ausführungen darauf, die einzelnen Verfeinerungsmechanismen in ihrer generellen Funktionsweise vorzustellen, statt einen Überblick über den gesamten Ablauf des Verfeinerungsvorgangs anzubieten.

4.2.2.1 Verfeinerung der modellierten Mechanik

In einem ersten Schritt der Mechanik-Verfeinerung wurden alle relevanten Bestandteile des erarbeiteten Systemmodells in die entsprechenden Bereiche des COLLADA-Formats transformiert. Dies beinhaltete im Einzelnen die Transformation der modellierten Bauteile (z.B. Förderband, Stempel) in den Geometriebereich, die Transformation von kinematischen Energie-Ports und -kanälen (z.B. des Förderbandes und des Material-Stoppers) in das Kinematik-Modell sowie die Transformation der Komponentenhierarchie in das visuelle Modell des COLLADA-Formats. Außerdem wurde für jede Komponente, die ein Bauteil im IMoMeSA-Modell besaß, ein eigener Starrkörper im Physik-Modell von COLLADA angelegt. Das generierte COLLADA-File konnte anschließend in das verwendete MCAD-Tool AutoCAD [51] eingelesen und dort bearbeitet werden. Wichtig ist an dieser Stelle zu erwähnen, dass die Bearbeitung in AutoCAD ausschließlich die Geometrien und Verbindungen betraf. Eine Veränderung an der Komponentenstruktur bzw. dem visuellen Modell musste gemäß dem spezifizierten Vorgehen stets im IMoMeSA-

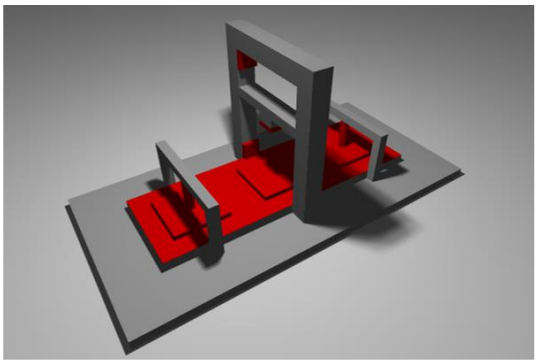
Modell durchgeführt werden, da strukturelle Änderungen dem gesamten interdisziplinären Entwicklungsteam zur Verfügung gestellt werden sollten, um Inkonsistenzen während des Verfeinerungsprozesses vermeiden zu können.

Nach dieser initialen Transformation galt es im Rahmen der Verfeinerung herauszufinden, für welche der modellierten Bauteile überhaupt eine Detaillierung notwendig war. Als erstes Beispiel für die Verfeinerung ist an dieser Stelle das Förderband aufgeführt. Wie bereits in Abschnitt 4.2.1.1 dargelegt, wurde für dieses in der Konzeption die vereinfachte Geometrie eines Quaders gewählt. Diese wurde in AutoCAD als Basis verwendet und zunächst hinsichtlich der tatsächlich einzuhaltenden Maße angepasst. Außerdem mussten an den Seiten des Förderbands Wände eingefügt werden, die für die Führung des Papiers auf dem Förderband verantwortlich sind.

In diesem Beispiel wurden die Bauteile einer bestehenden atomaren Komponente (Förderband) ausgetauscht, was der dritten und einfachsten Möglichkeit zur Verfeinerung einer Komponente entspricht (vgl. Abbildung 45). In einem zweiten Beispiel, welches bei der mechanischen Verfeinerung des Bedruckungsmoduls auftrat, musste stattdessen eine bereits zusammengesetzte Komponente um eine neue Unterkomponente erweitert werden. Im Konkreten musste zur Ablage des Papiers an der Endposition eine entsprechende Auffangvorrichtung konstruiert und an das Förderband angeflanscht werden. Für diese Aufgabe wurde zunächst eine neue Komponente „Auffangvorrichtung“ im IMoMeSA-Modell angelegt, um auch die übrigen Disziplinen über diese Änderung des Moduls zu informieren. Im IMoMeSA-Modell wurde für diese Komponente allerdings kein vereinfachtes Bauteil mehr modelliert, stattdessen wurde unmittelbar die aktualisierte Komponentenhierarchie in das visuelle Modell der COLLADA-Datei übertragen. Diese konnte anschließend erneut in AutoCAD eingelesen werden, wobei die Assembly-Struktur automatisch aktualisiert wurde. Innerhalb dieser Struktur konnte schließlich die Auffangvorrichtung konstruiert und in das gesamte MCAD-Modell eingebettet werden.

Auf vergleichbare Weise konnten die gesamten mechanischen Bestandteile des IMoMeSA-Modells verfeinert werden. Als Ergebnis entstand ein vollständiges MCAD-Modell des Bedruckungsmoduls, welches abschließend in Abbildung 59 der Grobgeometrie aus dem IMoMeSA-Modell gegenübergestellt wird.

(a) IMoMeSA Modell



(b) MCAD-Modell

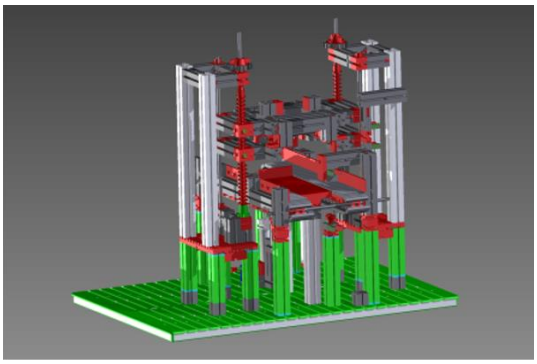


Abbildung 59: Gegenüberstellung des mechanischen Konzeptes nach Konzeption und Verfeinerung

4.2.2.2 Verfeinerung der modellierten Elektrik

Aufgrund der geringen Anzahl an Sensoren und Aktoren, die zur Realisierung des Bedruckungsmoduls benötigt wurden, war die Aufgabe der Schaltplan- bzw. ECAD-Modellierung von geringer Komplexität. Im Wesentlichen mussten bei der Elektrokonstruktion die tatsächlich zu verbauenden Sensoren und Aktoren identifiziert werden und mit den Ein- und Ausgangsklemmen der verwendeten B&R-Steuerung [52] verkabelt werden. Aus diesem Grund entschieden sich die Entwickler der Elektrik dazu, anstelle des vorgeschlagenen EPLAN-Formats eine einfachere Visualisierung des Schaltplans zu erstellen und dabei die Software MS Visio [53] einzusetzen.

Um dennoch eine Verfeinerung der Elektrik gemäß der IMoMeSA-Verfeinerungsmethodik durchzuführen, wurde die Transformation des Modells zumindest auf theoretischer Ebene durchgeführt. Um den grundsätzlichen Aufbau der Komponentenstruktur auch bei der vereinfachten Schaltplankonstruktion berücksichtigen zu können, wurde für jede der einzelnen elektrischen Komponenten (Antrieb, Lichtschranke, etc.) ein einzelnes Visio-Template (anstelle eines Makros) angelegt, um so den Schaltplan für jede Komponente separat ausarbeiten zu können. Lediglich die hierarchische Struktur ging bei dieser vereinfachten Modellierung verloren. Innerhalb jedes Templates wurden anschließend die elektrischen Energie- und Daten-Ports in entsprechende Ankerpunkte übertragen. Außerdem wurde in jedem Template standardmäßig eine Ein- und Ausgangsklemme sowie eine Spannungsversorgung und Erdung integriert, so dass die Verfeinerung in einfachsten Fall lediglich die Verknüpfung der Ankerpunkte mit den eben genannten Elementen beinhaltete.

Exemplarisch soll der Verfeinerungsvorgang am Beispiel einer der Lichtschranken des Förderbands vorgestellt werden. Abbildung 60a zeigt zunächst die entsprechende IMoMeSA-Komponente. Im Unterschied zur ursprünglichen Spezifikation besitzt diese neben dem Daten- und Material-Port auch einen Energie-Port, der in Abbildung 51 aus Übersichtsgründen nicht dargestellt wurde. Diese Komponente wurde anschließend in das Visio-Template transferiert, wobei der Daten- und der Energie-Port in entsprechende Ankerpunkte konvertiert wurden. Da der Sensor zu diesem Zeitpunkt noch nicht hinsichtlich seiner genauen elektrischen Ansteuerung betrachtet wurde, musste dieser Schritt in der Verfeinerung durchgeführt werden. Dadurch ergab sich, dass sowohl der Sender als auch der Empfänger eine Stromversorgung mit 24 Volt Spannung benötigen. Außerdem musste für den Sender eine zusätzlicher Ankerpunkt für die Erdung vorgesehen werden (vgl. Abbildung 60b). Schließlich konnte die Verknüpfung mit den übrigen Elementen des Templates durchgeführt werden, wodurch die Elektrokonstruktion für diese beispielhafte Komponente abgeschlossen war.

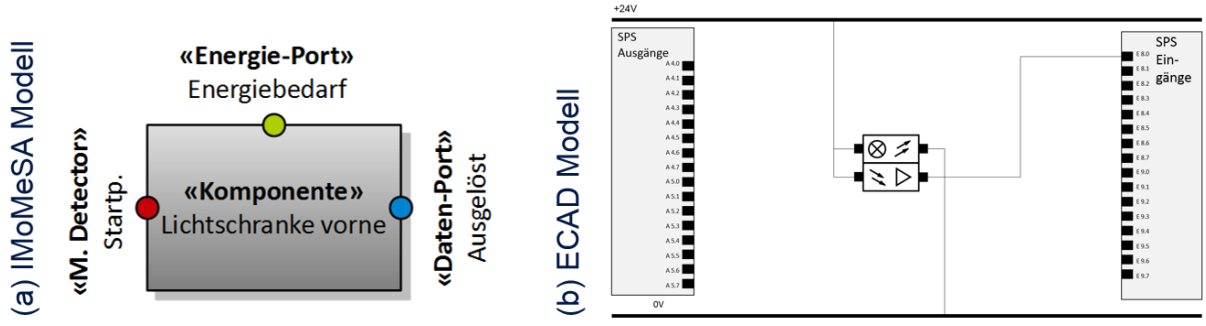


Abbildung 60: Verfeinerung der Elektrik am Beispiel einer Lichtschranke

4.2.2.3 Verfeinerung der modellierten Steuerungslogik

Analog zur Mechanik und Elektrik wurde bei der Verfeinerung der Steuerungslogik zunächst eine Transformation aller relevanten Bestandteile des IMoMeSA-Modells in das B&R Automation Studio durchgeführt. Im Einzelnen wurden die drei Steuerungs-Komponenten (Transport-, Bearbeitungs- und Gesamtsteuerung) in einzelne Funktionsblöcke transformiert sowie eine der IMoMeSA-Komponentenhierarchie entsprechende Programm-, Variablen- und Paketstruktur generiert. Es sei an dieser Stelle darauf hingewiesen, dass es sich bei den beiden verwendeten Timern um Standard-Komponenten der SPS-Entwicklung (TON) handelt, für die lediglich die Programme, aber keine eigenen Funktionsblöcke generiert werden mussten.

Als Beispiel für einen generierten Funktionsblock wird nachfolgend der Code der Steuerung der Transport-Komponente diskutiert (vgl. Abbildung 61). Dieser setzt sich aus einem Deklarationsteil und einem Rumpf zusammen. Im Deklarationsteil werden die einzelnen Daten-Ports der Steuerung (vgl. Abbildung 56) mit ihren bereits auf Modellebene definierten Datentypen eingetragen. Im Rumpf des generierten Funktionsblocks wird schließlich der modellierte Zustandsautomat in der Programmiersprache „Strukturierter Text“ (ST) abgebildet. Dabei werden die Zustände des Automaten (vgl. Abbildung 57) zu nummerierten Fällen einer „Switch-Case“-Anweisung, wobei die Namen der Zustände zur besseren Lesbarkeit als Kommentare den einzelnen Fällen zugeordnet werden. Innerhalb dieser Fälle werden die modellierten Aktionen (z.B. Starten des Antriebs) in Zuweisungen übersetzt. Schließlich werden Transitionen zu „If-endif“-Anweisungen transformiert, die einen Wechsel in der „Switch-Case“-Anweisung auslösen (z.B. Auslösen der Lichtschranke).

(a) Deklarationsteil

```
FUNCTION_BLOCK Steuerung
    VAR_INPUT
        Abfuehren : BOOL;
        Lichtschranke_vorne_unterbrochen : BOOL;
        Lichtschranke_hinten_unterbrochen : BOOL;
        Timer_Abgelaufen : BOOL;
        Quittiert : BOOL;
    END_VAR
    VAR_OUTPUT
        Zugefuehrt : BOOL;
        Fehlermeldung : BOOL;
        Antrieb_starten : BOOL;
        Ventil_oeffnen : BOOL;
        Timer_Start : BOOL;
        Timer_Zeit : TIME;
    END_VAR
    VAR
        State : INT;
    END_VAR
END_FUNCTION_BLOCK
```

(b) Rumpf (Ausschnitt)

```
FUNCTION_BLOCK Steuerung
    CASE State OF
        0: (*Warten*)
            IF Lichtschranke_vorne_unterbrochen  THEN
                Antrieb_starten             := TRUE;
                Ventil_oeffnen              := TRUE;
                Timer_Zeit                  := T#5s;
                Timer_Start                 := TRUE;
                State                       := 10;
            END_IF
        10: (*Zuführen*)
            IF Timer_Abgelaufen AND Endlagensensor_ausgeloest THEN
                Antrieb_starten             := FALSE;
                Ventil_oeffnen              := FALSE;
                Zugefuehrt                  := TRUE;
                State                       := 20;
            ELSIF Timer_Abgelaufen AND NOT Endlagensensor_ausgeloest THEN
                Antrieb_starten             := FALSE;
                Ventil_oeffnen              := FALSE;
                Fehlermeldung               := TRUE;
                State                       := 11;
            END_IF
```

Abbildung 61: Generierter Funktionsblock der Steuerung der Transportkomponente

Zur Instanziierung der generierten Funktionsbausteine sieht der IMoMeSA-Ansatz jeweils eigene Programme zur weiterführenden Kapselung der Funktionalität vor. Beispielhaft wird an dieser Stelle auf das generierte Programm der Steuerung der Transportkomponente eingegangen (vgl. Abbildung 62). In einem ersten Schritt des generierten Programms wird der zugehörige Funktionsblock aufgerufen und ausgeführt, was der zuvor erwähnten Instanziierung entspricht. Darüber hinaus ist das Programm für die Kommunikation mit anderen Modulen über eine globale Datenstruktur verantwortlich. Dafür werden einzelne Zuweisungen definiert und

ausgeführt, welche die im Funktionsblock deklarierten Variablen auslesen oder beschreiben. Beispielhaft wird an dieser Stelle die Kommunikation mit dem Timer näher beschrieben. Nach Ausführung des Funktionsblocks innerhalb eines SPS-Zyklus werden die Werte der Output-Variablen „Timer_Start“ und „Timer_Zeit“ des Funktionsblocks in die entsprechenden Variablen der globalen Datenstruktur geschrieben, wo sie von anderen Funktionsblöcken gelesen werden können. In umgekehrter Richtung wird der aktuelle Wert der Variable „Timer_abgelaufen“ von der globalen Datenstruktur ausgelesen und an den Funktionsblock weitergeleitet.

```
PROGRAM _CYCLIC

    Steuerung();

    Bedruckungsmodul.Transportmodul.Steuerung.Timer_Start              := Steuerung.Timer_Start;
    Bedruckungsmodul.Transportmodul.Steuerung.Timer_Zeit               := Steuerung.Timer_Zeit;
    Steuerung.Timer_Abgelaufen                                         := Bedruckungsmodul.Transportmodul.Timer.Timer_Abgelaufen;

    Steuerung.Lichtschranke_vorne_unterbrochen                         := Bedruckungsmodul.Transportmodul.Lichtschranke_vorne.unterbrochen;
    Steuerung.Lichtschranke_hinten_unterbrochen                        := Bedruckungsmodul.Transportmodul.Lichtschranke_hinten.unterbrochen;

    Bedruckungsmodul_Instanz.Transportmodul.Steuerung.Band_starten     := Steuerung.Band_starten;
    Bedruckungsmodul_Instanz.Transportmodul.Steuerung.Ventil_oeffnen   := Steuerung.Ventil_oeffnen;

    Steuerung.Abfuehren                                                := Bedruckungsmodul.Bearbeitungsmodul.Steuerung.Stempeln_beendet;
    Bedruckungsmodul_Instanz.Transportmodul.Steuerung.Zugefuehrt       := Steuerung.Zugefuehrt;

END_PROGRAM
```

Abbildung 62: Programm zur Instanziierung des Funktionsblocks

Wie bereits in den vorigen Abschnitten angedeutet, wird zur Kommunikation zwischen einzelnen Programmen eine globale Datenstruktur verwendet, die ebenfalls vollständig aus dem IMoMeSA-Modell generiert werden kann und grundsätzlich die IMoMeSA-Komponentenstruktur widerspiegelt. Realisiert wird diese Datenstruktur durch eine globale Variable, die vom komplexen Datentyp „Bedruckungsmodul“ ist. Dieser Typ wird in einer eigenen TYP-Datei definiert und setzt sich dabei aus den komplexen Datentypen „Transportkomponente“, „Bearbeitungskomponente“ und „Gesamtsteuerung“ zusammen (vgl. Abbildung 63). Analog zur IMoMeSA-Komponentenstruktur endet diese Zerlegung bei atomaren Komponenten, für die schließlich die Daten-Ports mit ihren Standard-Datentypen in entsprechende Variablen transformiert werden (z.B. Gesamtsteuerung).

```
TYPE
    Bedruckungsmodul :  STRUCT
        Gesamtsteuerung : Gesamtsteuerung;
        Bearbeitungskomponente : Bearbeitungskomponente;
        Transportkomponente : Transportkomponente;
    END_STRUCT;
    Gesamtsteuerung :   STRUCT
        Sensoren_aktivieren : BOOL;
        Aktoren_aktivieren : BOOL;
    END_STRUCT;
    Transportkomponente :   STRUCT
        Steuerung : Transportkomponente_Steuerung;
        Timer : Timer;
    END_STRUCT;
    Bearbeitungskomponente :    STRUCT
        Steuerung : Bearbeitungskomponente_Steuerung;
        Timer : Timer;
    END_STRUCT;
```

Abbildung 63: Ausschnitt der TYP-Datei zum Aufbau der globalen Datenstruktur

Aufbauend auf der initialen Erstellung des gesamten B&R-Steuerungsprojektes konnte die eigentliche Verfeinerung des generierten Codes durchgeführt werden. Wegen der geringen Komplexität der Steuerungsaufgabe konnte die geforderte Funktionalität bereits vollständig im IMoMeSA-Modell über einzelne Steuerungskomponenten bzw. deren Zustandsautomaten realisiert werden. Aus diesem Grund konnte der generierte Code ohne weitere Adaptionen auf die Steuerung übertragen werden. Die Aufgabe der Verfeinerung reduzierte sich deswegen auf die Verknüpfung der physikalischen Ein- und Ausgänge mit den entsprechenden Klemmen der B&R-Steuerung gemäß den erstellten Schaltplänen (vgl. Abschnitt 4.2.2.2) sowie die Zuweisung der erstellten Programme zu einzelnen Task-Scheiben der SPS, um die jeweiligen Zykluszeiten festzulegen.

4.3 Bewertung der modellbasierten Entwicklungsmethodik

Durch die Anwendung der IMoMeSA-Entwicklungsmethodik an dem Fallbeispiel des zuvor beschriebenen Bedruckungsmoduls konnten die grundsätzlichen Stärken und Optimierungspotenziale des Ansatzes herausgestellt werden. Diese werden in Abschnitt 4.3.1 getrennt für die Phasen der Konzeption und Verfeinerung vorgestellt. Darauf aufbauend wird der wissenschaftlich-technische sowie der wirtschaftliche Nutzen des IMoMeSA-Ansatzes insbesondere für kleine und mittlere Unternehmen (KMU) analysiert (Abschnitt 4.3.2), bevor schließlich die Übertragbarkeit der Forschungsergebnisse in das industrielle Umfeld diskutiert wird (Abschnitt 4.3.3).

4.3.1 Stärken und Schwächen der Entwicklungsmethodik

Wie bereits in Abschnitt 4.1 erläutert wurde, sollten die Stärken und Schwächen des IMoMeSA-Ansatzes insbesondere vor dem Hintergrund einer zukünftigen industriellen Anwendung aufgezeigt werden. Zu diesem Zweck wurden sowohl für die Konzeption als auch die Verfeinerung die folgenden Fragen beantwortet:

1.) Welche Inhalte der Modellierungstechnik können einen realen Entwicklungsprozess im Werkzeugmaschinenbau zielführend unterstützen? (Stärken)
2.) Welche der erarbeiteten Modellierungsaspekte werden den aktuellen Anforderungen dieser Branche eher nicht gerecht? (Schwächen)
3.) Welche weiteren Aspekte müssen noch in die Modellierungstechnik integriert werden, um eine langfristige Etablierung eines modellbasierten Entwicklungsansatzes im Werkzeugmaschinenbau sicherzustellen? (Ausblick)

4.3.1.1 Konzeption

Um die oben gestellten Fragen für die Konzeption beantworten zu können, wurden die erarbeiteten Forschungsergebnisse mit besonderem Fokus auf das Fallbeispiel kontinuierlich mit den Unternehmen des Werkzeugmaschinenbaus diskutiert (vgl. Abschnitt 4.1). Die Ergebnisse dieser Diskussionen sind in Tabelle 1 zusammengefasst, in der die einzelnen Aspekte der modellbasierten Entwicklungsmethodik in die Felder *Stärken*, *Schwächen* und *Ausblick* unterteilt sind.

Tabelle 1: Bewertung der modellbasierten Entwicklungsmethodik für die Konzeption

Stärken	- Spezifikation von Abnahmetests - Einfache Aufnahme und Strukturierung von Kundenanforderungen - Aufbau einer modularen Komponentenstruktur - Spezifikation von Prozessen und Verhalten - Möglichkeiten zur interdisziplinären Qualitätssicherung - Explizite Berücksichtigung von möglichem Fehlverhalten
Schwächen	- Notwendigkeit der vollständigen Formalisierung des Systemmodells - Komplexe Modellierung von Bauteilen - Qualitätssicherungsmaßnahmen zu umfangreich
Ausblick	- Einfache und informelle Prinzipskizzen - Einsatz eines mechatronischen Baukastensystems - Unterstützung des Änderungs- und Variantenmanagements

Als eine zentrale Stärke des IMoMeSA-Ansatzes wurde im Rahmen der Evaluation zunächst die Möglichkeit der einfachen Kundeneinbeziehung herausgestellt. Besonders hervorzuheben ist hierbei die Möglichkeit, spätere Abnahmetests von Entwicklungsbeginn an über Szenarien modellieren und diese als Testfälle für eine virtuelle Konzeptabsicherung nutzen zu können. Außerdem ermöglicht es der IMoMeSA-Ansatz, Kundenanforderungen zunächst ohne großen Aufwand über Karteikarten aufzunehmen und anschließend entsprechende Mechanismen zur Formalisierung anzubieten, die eine direkte Überprüfung ermöglichen. Als ein weiterer wichtiger Aspekt in der frühen Entwicklungsphase wurde von den Unternehmen der Aspekt der Zerlegung des mechatronischen Systems in gekapselte und wiederverwendbare Module genannt. Dieser Aspekt wird im IMoMeSA-Ansatz explizit durch die hierarchische Modellierungstechnik mit Komponenten und Ports unterstützt. Neben dem reinen Aufbau des zu entwickelnden Systems wurde als weitere Stärke die Möglichkeit herausgestellt, Prozesse und Verhalten auf einfache Weise über Zustandsautomaten spezifizieren zu können. Bedingt durch die kontinuierlich steigende Komplexität bei mechatronischen Systemen wird es zukünftig notwendig sein, das Verhalten einzelner Systemelemente zunächst lösungsneutral beschreiben zu können, um ein interdisziplinäres Verständnis aufzubauen. Als eine weitere Stärke des IMoMeSA-Ansatzes wurde die Möglichkeit zur interdisziplinären Qualitätssicherung genannt, um unterschiedliche Konzepte gegeneinander evaluieren zu können oder um die Konsistenz von erarbeiteten Teillösungen sicherzustellen. Schließlich sieht der IMoMeSA-Ansatz die explizite Berücksichtigung von Verhaltensfehlern sowie deren Behandlung vor. Dieser Aspekt wurde von den Unternehmen ebenfalls als wichtig eingestuft, da die Fehlersuche und -behandlung derzeit zumeist erst während der Inbetriebnahme oder sogar erst in der Hochlaufphase durchgeführt wird, was sowohl zu Verzögerungen als auch zu einer generellen Unzufriedenheit des Kunden führen kann.

Andererseits zeigte sich in der Evaluierung, dass einige Aspekte der Modellierungstechnik als weniger zielführend für die Konzeption von mechatronischen Systemen identifiziert wurden. Im Speziellen wurde es als kritisch angesehen, dass die vollständige Formalisierung von Systemmodellen (notwendig zur Anwendung der Simulation) einen erheblichen Modellierungsaufwand erfordert, der aufgrund des bestehenden Zeit- und Kostendrucks üblicherweise nicht vorhanden ist. Außerdem wurde die Geometriemodellierung (insbesondere hinsichtlich der unterschiedlichen Material-Ports) als zu komplex für die frühe Entwicklungsphase bewertet. Vielmehr sollten in dieser Phase einfacher zu modellierende Bauteile, Achsen und Bewegungen zur Anwendung kommen, wie es bspw. vom „mechatronischen Sketching“ [54] bekannt ist. Schließlich wurden auch die Qualitätssicherungsmaßnahmen als zu umfangreich angesehen. Während grundsätzlich die Möglichkeit zur Qualitätssicherung als positiver Aspekt herausgestellt wurde, werden sowohl die Maßnahmen des Unit-Testens und der kontinuierlichen Integration als auch die Regelmäßigkeit der durchzuführenden Absicherung innerhalb des definierten Vorgehens als nicht praktikabel für den Werkzeugmaschinenbau angesehen.

Neben der Bewertung des aktuellen Standes des IMoMeSA-Ansatzes konnten im Rahmen der Evaluation ebenfalls Aspekte identifiziert werden, welche die Konzeption von mechatronischen System bei einer modellbasierten Entwicklung unterstützen können, bisher aber noch nicht berücksichtigt wurden. Unter diesen ist zunächst das Anfertigen von einfachen und informellen Prinzipskizzen zu nennen. Solche Skizzen werden in der frühen Entwicklungsphase oft verwendet, um verschiedenste Aspekte eines mechatronischen Systems zwischen den beteiligten Disziplinen zu diskutieren. Darüber hinaus kann der Einsatz eines mechatronischen Baukastensystems den gesamten Entwicklungsansatz unterstützen, da die Wiederverwendung von Komponenten (z.B. über Templates) mittlerweile essenziell ist, um Entwicklungszeiten und -kosten reduzieren zu können. Ein ebenfalls wichtiger Aspekt, der bei einer Weiterentwicklung des IMoMeSA-Ansatzes berücksichtigt werden sollte, ist die Unterstützung des Änderungs- und Variantenmanagements. Da der Trend zu einer kundenspezifischen Fertigung immer mehr an Bedeutung gewinnt, sind die Erstellung von Varianten und die unmittelbare Reaktion auf sich ändernde Kundenanforderungen wichtige Aspekte, die zukünftig durch eine modellbasierte Entwicklungsmethodik unterstützt werden müssen.

4.3.1.2 Verfeinerung

Analog zur Konzeption wurden auch bei der Verfeinerung die Ergebnisse des Fallbeispiels verwendet, um die Stärken und Schwächen der modellbasierten Entwicklungsmethodik mit dem projektbegleitenden Ausschuss zu diskutieren. Die Ergebnisse dieser Diskussion sind wie im vorangegangenen Abschnitt in Tabelle 2 zusammengefasst und in *Stärken*, *Schwächen* und *Ausblick* unterteilt.

Tabelle 2: Bewertung der modellbasierten Entwicklungsmethodik für die Verfeinerung

Stärken	- Integration bestehender Entwicklungswerkzeuge - Wiederverwendung von Lösungen aus der Konzeption - Förderung des interdisziplinären Austausches - Durchgängigkeit der Qualitätssicherungsmaßnahmen - Systematisches Vorgehen mit inkrementeller Verfeinerung
Schwächen	- Komplizierte Elemente zur Modellierung der Simulation Units - Umständlicher Ansatz zur Co-Simulation - Unzureichende Evaluierung am Fallbeispiel
Ausblick	- Explizite Berücksichtigung der Fluidik (z.B. Hydraulik, Pneumatik) in der Modellierung - Verknüpfung mit ERP-Systemen

Als eine der zentralen Stärken des IMoMeSA-Ansatzes in der Verfeinerung wurde der bereits bei den Anforderungen postulierte Aspekt der Integration bestehender Entwicklungswerkzeuge genannt. Insbesondere bei den Deployment Units werden ausschließlich bereits etablierte Werkzeuge eingesetzt, um bereits vorhandenes Know-How der Entwickler ideal nutzen zu können und gleichzeitig den Mehraufwand bei der Einführung dieses Ansatzes im industriellen Umfeld auf ein Minimum zu reduzieren. Eng verknüpft mit diesem Punkt ist der Aspekt der Wiederverwendung von Lösungen aus der Konzeption. Durch die erarbeiteten Modelltransformationen können alle für die Verfeinerung relevanten Aspekte des Systemmodells automatisch in die einzelnen Entwicklungswerkzeuge übertragen werden. Dies kann die Akzeptanz für einen modellbasierten Entwicklungsansatz erheblich steigern, da alle Aufwände, die während der Konzeption zum Aufbau eines Systemmodells notwendig sind, durch Einsparungen in späteren Phasen gerechtfertigt werden können. Als eine weitere Stärke der modellbasierten Entwicklungsmethodik wurde herausgestellt, dass der Ansatz den interdisziplinären Austausch auch in der Verfeinerungsphase fördert. Dies wird insbesondere durch den durchgängigen Einsatz des Systemmodells ermöglicht, mit dem alle Teilergebnisse aus den einzelnen Disziplinen bzw. Werkzeugen während der gesamten Verfeinerung synchron gehalten werden müssen. Zu diesem Aspekt trägt auch bei, dass es der IMoMeSA-Ansatz ermöglicht, die bereits aus der Konzeption bekannten Qualitätssicherungsmaßnahmen über eine Co-Simulation nutzen zu können. Durch die Anwendung von Kompilierung, Simulation, Unit-Testen und kontinuierlicher Integration können Inkonsistenzen rechtzeitig erkannt und Entwicklungsentscheidungen über Disziplinen hinweg stetig abgesichert werden. Als eine weitere Stärke des Ansatzes wurde schließlich der definierte Workflow insbesondere mit der Möglichkeit zur inkrementellen Verfeinerung einzelner Komponenten genannt. Dadurch, dass die Arbeiten in der Verfeinerung in kleine Teilaufgaben heruntergebrochen und deren Ergebnisse am Ende eines jeden Schrittes im interdisziplinären Gesamtkontext abgesichert werden, können Fehler rechtzeitig erkannt und kostspielige Iterationsschleifen vermieden werden.

Allerdings wurde in der Evaluation auch aufgezeigt, dass der IMoMeSA-Ansatz in der Verfeinerung derzeit verschiedene Schwachstellen besitzt. Unter diesen sind zunächst die Simulation Units zu nennen. Die einzelnen Elemente stützen sich jeweils auf komplexe Formalismen oder Modellierungssprachen, die zudem im Werkzeugmaschinenbau aktuell kaum zum Einsatz kommen. Dies erschwert eine langfristige Etablierung und widerspricht insbesondere der Forderung, den Aufwand zur Einführung der modellbasierten Entwicklungsmethodik möglichst gering zu halten. Als weiterer Schwachpunkt des IMoMeSA-Ansatzes wurde der skizzierte Weg zur Co-Simulation identifiziert. Dieser stützt sich im Wesentlichen auf das Functional Mockup Interface, welches für eine Anwendung den zeitaufwändigen Aufbau eigener FMI-DLLs benötigt. Zudem ist das FMI zum aktuellen Zeitpunkt noch nicht im industriellen Umfeld etabliert und wurde auch im Rahmen von IMoMeSA lediglich als gangbarer Weg zur Qualitätssicherung identifiziert, aber nicht umgesetzt und auch nicht angewandt. Somit können aktuell keine detaillierten Aussagen dazu getroffen werden, ob tatsächlich eine Kopplung von allen für den IMoMeSA-Ansatz benötigten Simulatoren möglich ist. Eine weitere Schwachstelle, die im Kontext der Verfeinerung aufzulisten ist, betrifft schließlich die unzureichende Anwendung der einzelnen Modellierungsaspekte der Verfeinerung im Rahmen des Fallbeispiels. Insbesondere konnten die Simulation Units gar nicht und die Deployment Units mit den zugehörigen Modelltransformationen nur partiell angewendet werden. Der Co-Simulationsansatz konnte aus den eben genannten Gründen ebenfalls nicht evaluiert werden. Deswegen besitzen die Aussagen zum praktischen Nutzen sowie zur industriellen Anwendbarkeit der Entwicklungsmethodik in der Verfeinerung derzeit nur eine eingeschränkte Aussagekraft.

Auch in der Verfeinerung konnten schließlich Potenziale aufgezeigt werden, die bei einer Erweiterung der Modellierungstechnik zu berücksichtigen sind. Unter diesen ist zunächst die Erweiterung der Modellierungstechnik um Aspekte der Fluidik zu nennen. Während in der Konzeption die Berücksichtigung von Fluiden noch durch die allgemeine Modellierung von Ports und Kanälen funktionieren kann, bedarf es in der Verfeinerung expliziter Mechanismen, um fluidische Phänomene modellieren und simulieren zu können oder spezifische externe Werkzeuge anzubinden. Des Weiteren wurde von den Unternehmen auch der Bedarf aufgedeckt, eine Integration mit ERP-Werkzeugen zu ermöglichen, um so bspw. Stücklisten automatisch zu generieren und somit den Vorgang des Bestellwesens besser im Entwicklungsprozess zu verankern.

4.3.2 Wissenschaftlich-technischer Mehrwert und wirtschaftlicher Nutzen

Das Themenfeld der modellbasierten Entwicklung mechatronischer System ist bereits seit einigen Jahren im Fokus von Wissenschaft und Industrie. Neben unterschiedlichen Forschungsansätzen (wie bspw. [6][20][55]) wurden auch erste Engineering-Werkzeuge wie das Siemens Team Center [56], der Mechatronics Concept Designer [29] oder die EPLAN Engineering Configuration [57] entwickelt und eingeführt. Die im Rahmen des IMoMeSA-Projektes erarbeitete Entwicklungsmethodik baut auf diesen Vorarbeiten auf und erweitert die aktuellen Ansätze um bislang nicht betrachtete Aspekte, die allerdings im Rahmen eines mechatronischen Entwicklungsprozesses von zentraler Bedeutung sind. Diese Aspekte werden in den nächsten Abschnitten zunächst einzeln vor dem Hintergrund ihres wissenschaftlich-technischen Mehrwerts beschrieben. Abschließend wird der wirtschaftliche Nutzen der gesamten Modellierungstechnik (insbesondere für kleine und mittlere Unternehmen) dargestellt.

Als erster Aspekt, der einen wissenschaftlich-technischen Mehrwert im Vergleich zu bestehenden Lösungen bietet, ist das *Anforderungsmanagement* zu nennen. In aktuellen modellbasierten Ansätzen wird das Anforderungsmanagement entweder gar nicht oder nur als losgelöster Bestandteil des Entwicklungsprozesses betrachtet, was zu Reibungsverlusten führt und insbesondere die Durchgängigkeit erschwert. Der im Rahmen von IMoMeSA erarbeitete integrative Ansatz ermöglicht es hingegen, Anforderungen parallel zum Aufbau der mecharonischen Systemstruktur zu definieren und mit einzelnen Komponenten oder Zuständen zu verknüpfen. Insbesondere die Möglichkeit der Formalisierung von Anforderungen wurde im Umfeld der mechatronischen Entwicklung bisher nicht betrachtet. Die mittels Szenarien, Monitoren, Bedingungen und Qualitäten bereitgestellten Mechanismen ermöglichen eine intuitive Formalisierung von Abnahmetests, geforderten Funktionen und weiteren Kundenanforderungen. Die interdisziplinäre Qualitätssicherung bietet schließlich die Möglichkeit, die Implementierung ohne weitere Aufwände auf eine Einhaltung aller Anforderungen zu überprüfen.

Einen zweiten Entwicklungsbereich, in dem durch den IMomeSA-Ansatz ein Mehrwert geschaffen werden konnte, bildet das *Fehlermanagement*. Im wissenschaftlichen wie industriellen Umfeld spielt die Behandlung von Fehlern oder die Absicherung gegen das Auftreten von Fehlern bereits seit langem eine wichtige Rolle. So existieren bereits zahlreiche Normen, welche die Sicherheit von Maschinen und Anlagen in Bezug auf mögliche Fehler vorschreiben (z.B. [58][59]). Darüber hinaus gibt es verschiedene Methoden, die es ermöglichen, potenzielle Fehler in technischen Systemen zu erkennen und hinsichtlich ihrer Kritikalität zu bewerten sowie geeignete Gegenmaßnahmen abzuleiten (z.B. FMEA [60] oder Fehlerbaumanalyse [61]). Der IMoMeSA-Ansatz baut auf diesen Methoden auf und erweitert sie insbesondere durch die Einbettung in die erarbeitete Modellierungstechnik sowie die feste Verankerung im Entwicklungsprozess im Sinne eines systematischen Vorgehens. Gerade im Bereich der modellbasierten Entwicklung gibt es aktuell noch keine Ansätze, die eine explizite Modellierung von Verhaltensfehlern mechatronischer Systeme ermöglichen. Im IMoMeSA-Ansatz ist das Fehlermanagement hingegen ein integraler Bestandteil des Entwicklungsprozesses, was zu einer Steigerung der Robustheit mechatronischer Systeme bereits vor ihrem physikalischen Aufbau führt. Durch die formale Modellierung von Fehlern können zudem die Auswirkungen von Verhaltensfehlern in der Qualitätssicherung simuliert und analysiert werden. Diese Möglichkeit bieten aktuelle Methoden zur Fehlersuche und -behebung bislang nicht.

Des Weiteren bieten auch die einzelnen *Qualitätssicherungsmaßnahmen*, die im Rahmen von IMoMeSA erarbeitet wurden, einen wissenschaftlich-technischen Mehrwert. Die Simulation etabliert sich im Maschinen- und Anlagenbau zwar gerade als Werkzeug zur Qualitätssicherung, allerdings kann diese in bisherigen Ansätzen nicht zur formalen Überprüfung eines korrekten Systemverhaltens mittels Szenarien, etc. verwendet werden. Darüber hinaus definiert der IMomeSA-Ansatz die Maßnahmen des Unit-Testens und der kontinuierlichen Integration, die im Maschinen- und Anlagenbau bisher noch überhaupt nicht betrachtet wurden. Im Software Engineering ermöglichen diese Maßnahmen jedoch bereits seit längerem eine zielgerichtete Analyse von Systemen und eine kontinuierliche Zielüberwachung in Entwicklungsprojekten. Vor dem Hintergrund einer steigenden Komplexität ist zu erwarten, dass

vergleichbare Ansätze zukünftig auch im Maschinen- und Anlagenbau eine wichtige Rolle einnehmen werden. Hier bietet der IMoMeSA-Ansatz eine ideale Ausgangsbasis, um diese Mechanismen langfristig zu etablieren.

Einen wissenschaftlich-technischen Mehrwert bieten ebenfalls die im IMoMeSA-Ansatz vorgesehenen Möglichkeiten zur *Integration etablierter Entwicklungswerkzeuge* im Rahmen der Verfeinerung. Während vergleichbare modellbasierte Entwicklungsansätze die Zusammenarbeit mit digitalen Werkzeugen entweder gar nicht adressieren oder sich auf die Kommunikation mit wenigen eingeschränkten Tools konzentrieren, bietet der IMoMeSA-Ansatz eine zentrale Plattform, um möglichst viele rechnergestützte Entwicklungsschritte zu vereinfachen und zu synchronisieren. Zudem wurde der IMoMeSA-Ansatz so aufgebaut, dass er zukünftig im Sinne eines modularen Baukastensystems um weitere Entwicklungswerkzeuge erweitert werden kann, was insbesondere durch die Verwendung des offenen Functional Mockup Interface zur Co-Simulation ermöglicht wird. In diesem Kontext sind als wissenschaftlicher Mehrwert schließlich auch die eingeführten Modelltransformationen zu nennen. Der Mehrwert liegt dabei allerdings nicht auf den erarbeiteten Algorithmen zur Modelltransformation, da technologisch gesehen Modelltransformationen bereits seit längerem zum Stand der Technik zählen. Vielmehr liegt der geschaffene Mehrwert in der systematischen Anwendung der einzelnen Transformation im Rahmen des Entwicklungsprozesses, um den durch die Erarbeitung eines zentralen Systemmodells entstehenden Mehraufwand weitestgehend zu minimieren.

Neben dem oben dargestellten wissenschaftlich-technischen Mehrwert, kann die erarbeitete Entwicklungsmethodik auch einen *wirtschaftlichen Nutzen* für industrielle Anwender generieren. Durch die Nutzung des IMoMeSA-Ansatzes werden Unternehmen dazu befähigt, einzelne Entwicklungsbereiche (Systementwurf, Anforderungsmanagement, Konstruktion, SW-Entwicklung, etc.) aufeinander abzustimmen und dadurch Reibungsverluste auf ein Minimum zu reduzieren. Darüber hinaus ermöglicht die frühzeitige Synchronisation von allen beteiligten Entwicklern einen interdisziplinären Erkenntnisgewinn, der dazu führt, dass Fehler noch in frühen Phasen erkannt und langwierige Iterationsschleifen somit vermieden werden können. Weiterhin können durch die Mechanismen zur Modelltransformation viele Entwicklungsmodelle, wie zum Beispiel MCAD-Modelle, automatisiert abgeleitet werden, was eine zusätzliche Einsparung an Entwicklungszeit mit sich bringt. Neben der zeitlichen Komponente besitzt der IMoMeSA-Ansatz auch das Potenzial, die Qualität der zu entwickelnden Systeme zu verbessern. Dazu tragen insbesondere die entwickelten Konzepte zur Integration des Anforderungs- und Fehlermanagements in den Entwicklungsprozess bei. Auf der einen Seite können durch die erarbeitete Methodik Anforderungen über die gesamte Entwicklungsdauer berücksichtigt werden. Dies führt in der Regel dazu, dass das entwickelte System ideal an den vom Kunden geforderten Eigenschaften ausgerichtet ist und somit zu einer hohen Kundenzufriedenheit führt. Auf der anderen Seite besitzen mechatronische Systeme, die mittels des IMomeSA-Ansatzes entwickelt wurden, eine große Robustheit gegenüber möglichen Verhaltensfehlern, da diese bereits während der Konzeption modelliert und behandelt werden können. Dies ermöglicht es, die sonst erst während der Inbetriebnahme oder der Hochlaufphase auftretenden Fehler a priori ausschließen zu können und die anfänglichen Stillstandzeiten beim Kunden auf ein Minimum zu reduzieren.

Insbesondere für KMU, die mit ihrem innovativen Charakter häufig zu den Weltmarktführern zählen, ist die Optimierung ihrer Entwicklungsprozesse mittels der oben beschriebenen Potenziale von entscheidender Bedeutung. Der in IMoMeSA entwickelte, modellbasierte Ansatz kann dazu beitragen, die Voraussetzungen zu schaffen, dass qualitativ hochwertige Systeme der Produktionstechnik möglichst früh und ausgereift auf den Markt gebracht werden können. Durch die branchenübergreifende Gültigkeit und Anwendbarkeit der Forschungsergebnisse kommen diese einer Vielzahl von Unternehmen zugute. Hauptsächlich profitieren können Unternehmen, die komplexe Anlagen und Maschinen kundenspezifisch anfertigen und somit große Aufwände in die Entwicklung ihrer Systeme investieren müssen.

4.3.3 Übertragbarkeit der Forschungserkenntnisse

In einem letzten Schritt der Evaluierung werden Möglichkeiten und Maßnahmen aufgezeigt, um die erarbeiteten Ergebnisse in das industrielle Umfeld transferieren zu können. Im Rahmen des Forschungsprojektes wurde eine modellbasierte Entwicklungsmethodik aufgebaut und prototypisch in zwei Werkzeugen, dem „IMoMeSA Modeller“ und dem „IMoMeSA Analyzer“ umgesetzt. Diese Werkzeuge eignen sich derzeit jedoch nicht für eine unmittelbare Anwendung im industriellen Umfeld, da sie lediglich dem Zweck der Evaluierung dienten und deswegen weder vollständig umgesetzt noch auf eine intuitive Bedienung hin ausgerichtet wurden. Vor diesem Hintergrund gibt es somit grundsätzlich zwei Möglichkeiten, um ein Entwicklungswerkzeug zur Anwendung des IMoMeSA-Ansatzes aufzubauen. Auf der einen Seite kann der bisherige Quellcode als Basis verwendet werden, um die beiden oben genannten Werkzeuge bis zu einer Marktreife weiterzuentwickeln. Hierfür werden bei dem derzeitigen Entwicklungsstand ein Zeitraum von einem Jahr sowie ein Entwicklungsaufwand von ca. drei Personenjahren geschätzt. Auf der anderen Seite besteht die Möglichkeit, dass die erarbeitete Methodik in bereits etablierte Werkzeuge zur modellbasierten Entwicklung mechatronischer Systeme integriert wird. Im Rahmen dieses Forschungsprojektes wurden dazu die entsprechenden Werkzeuge der Firma Siemens (TeamCenter und Mechatronics Concept Designer) hinsichtlich einer möglichen Integrierbarkeit der einzelnen Aspekte des IMoMeSA-Ansatzes analysiert. In diesem Zuge konnten sowohl grundsätzliche Parallelen als auch mögliche Anknüpfungspunkte identifiziert werden, die eine reibungslose Integration der zentralen Aspekte der Entwicklungsmethodik ermöglichen.

Neben der notwendigen Ausarbeitung eines Werkzeuges zur Realisierung der in IMoMeSA erarbeiteten Modellierungstechnik ist es auch notwendig die Akzeptanz für den gesamten Ansatz im Maschinen- und Anlagenbau zu verbessern. Im Rahmen des Forschungsprojektes wurde dazu der Ansatz an dem Beispiel des Bedruckungsmoduls angewendet und evaluiert. Dieses Beispiel ist in seiner Komplexität jedoch eingeschränkt und erlaubt somit nur bedingt Aussagen zur industriellen Anwendbarkeit der modellbasierten Entwicklungsmethodik. Deswegen wird es als nächster wichtiger Schritt angesehen, den IMoMeSA-Ansatz an einem realen Fallbeispiel aus dem Maschinen- oder dem Anlagenbau zu erproben und dadurch für den Einsatz in der Industrie zu qualifizieren. Die Wichtigkeit dieser Maßnahme wurde auch vom projektbegleitenden Ausschuss bestätigt, so dass von den Projektbearbeitern bereits die ersten Schritte eingeleitet werden konnten, um diese weiterführende Evaluierung und Qualifizierung durchzuführen. Das geplante Vorgehen sieht es vor, im Rahmen eines

VDW-eigenfinanzierten, halbjährigen Projektes eine spezielle Komponente einer Werkzeugmaschine bei einem der Unternehmen des projektbegleitenden Ausschuss mittels des IMoMeSA-Ansatzes zu entwickeln und daraus Einsatzpotenziale für eine industrielle Anwendung zu identifizieren sowie konkrete Handlungsempfehlungen abzuleiten. In einer zweiten Stufe können diese Empfehlungen schließlich umgesetzt werden, um den IMoMeSA-Ansatz hinsichtlich der konkreten Anforderungen des Werkzeugmaschinenbaus zu optimieren und dadurch langfristig im industriellen Umfeld zu etablieren.

5 Zusammenfassung und Ausblick

Im Rahmen des Forschungsprojektes IMoMeSA wurde eine Entwicklungsmethodik erarbeitet, die es ermöglicht, mechatronische Systeme von einer ersten Produktidee bis zu einem virtuellen Prototyp modellbasiert zu entwickeln. Nach einer kurzen Motivation für die Anwendung modellbasierter Entwicklungsprinzipien im Maschinen- und Anlagenbau wurden die relevanten Vorarbeiten aus dem AiF-Forschungsprojekt „AutoVIBN“ vorgestellt. Die dort erarbeitete Modellierungstechnik bildete die Basis für die Arbeiten im IMoMeSA, weswegen die zentralen Konzepte dieses Ansatzes kurz beschrieben wurden. Darauf aufbauend wurde die konkrete Zielstellung des Projektes mit den jeweiligen Teilaufgaben zur Erweiterung des AutoVIBN-Ansatzes eingeführt. Im Einzelnen sollten die Ergebnisse aus dem Projekt AutoVIBN in den Bereichen Anforderungs- und Fehlermanagement, Modelltransformationen sowie der Integration bestehender Entwicklungswerkzeuge erweitert werden.

Aufbauend auf dieser Einführung wurden die Ergebnisse der durchgeführten Anforderungsanalyse dargestellt. Im Rahmen dieser Analyse wurde zunächst ein generischer Entwicklungsprozess aufgenommen, der den allgemeinen Entwicklungsablauf in einzelne Phasen und Aktivitäten untergliedert. Im Rahmen dieser Prozessaufnahme konnten bereits erste Schwachstellen identifiziert werden, die durch einen modellbasierten Entwicklungsansatz verbessert werden können. Darauf aufbauend wurde skizziert, welche digitalen Werkzeuge aktuell im Werkzeugmaschinenbau zum Einsatz kommen, um so Aussagen zu ermöglichen, inwiefern bestehende Entwicklungswerkzeuge sinnvoll in einen ganzheitlichen modellbasierten Ansatz integriert werden können. Schließlich wurden für die Bereiche des Anforderungs- und des Fehlermanagements jeweils eigene Übersichten bzw. Klassifikationen erstellt, die es ermöglichen, das gesamte Spektrum an Anforderungen bzw. Verhaltensfehlern von Werkzeugmaschinen abzubilden.

Diese Vorabreiten wurden beim Aufbau der modellbasierten Entwicklungsmethodik ebenso kontinuierlich berücksichtigt wie die eingangs formulierten Projektziele. Die so erarbeitete Methodik sieht grundsätzlich eine Unterteilung des Entwicklungsprozesses in die Phasen Konzeption und Verfeinerung vor. Für die erstgenannte Phase wurden zunächst die einzelnen Inhalte der Modellierungstechnik auf Meta-Modellebene und unter Verwendung der Unified Modeling Language (UML) vorgestellt. Darauf aufbauend wurden die einzelnen Qualitätssicherungsmaßnahmen eingeführt, die es ermöglichen, modellierte mechatronische Systeme auf syntaktische und semantische Korrektheit zu überprüfen. Für den Umgang mit der erarbeiteten Modellierungstechnik wurde des Weiteren ein systematisches Entwicklungsvorgehen vorgestellt, welches die Konzeption zunächst hinsichtlich der Modellierung des Normal- und des Fehlverhaltens unterteilt. Die Modellierung des Normalverhaltens umfasst dabei den Aufbau einer mechatronischen Komponentenstruktur und die anschließende Integration erarbeiteter Module zu einem funktionsfähigen Gesamtsystem. Das so entstandene mechatronische Konzept des Gutablaufs kann anschließend mittels eines dreistufigen Vorgehens um Fehlerbehandlungsmechanismen erweitert und dadurch in seiner Robustheit gesteigert werden. In der Verfeinerung wird schließlich das entstandene mechatronische System unter Verwendung etablierter Entwicklungswerkzeuge detailliert und bis zu einem virtuellen Prototyp weiterentwickelt. Analog zur Konzeption wurden im Rahmen des Abschlussberichtes zunächst die entsprechenden

Erweiterungen der Modellierungstechnik, die sog. Simulation und Deployment Units, vorgestellt. Anschließend wurden die einzelnen Modelltransformationen beschrieben, die im Rahmen von IMoMeSA erarbeitet wurden, um einen automatisierten Transfer der Daten des Systemmodells in die jeweiligen Entwicklungswerkzeuge zu ermöglichen. Darauf aufbauend wurde eine Möglichkeit aufgezeigt, wie die bereits in der Konzeption genutzten Qualitätssicherungsmaßnahmen auch in der Verfeinerung über eine Co-Simulation verwendet werden können. In einem letzten Schritt wurde auch für die Verfeinerung ein systematisches Entwicklungsvorgehen vorgeschlagen, welches über einen inkrementellen Ansatz eine komponentenweise Verfeinerung des erarbeiteten Systemmodells vorsieht.

Um die industrielle Anwendbarkeit der Modellierungstechnik evaluieren zu können, wurde die erarbeitete Entwicklungsmethodik während des Forschungsprojektes kontinuierlich an dem Fallbeispiel eines Bedruckungsmoduls einer miniaturisierten Verpackungsanlage gespiegelt. Im Abschlussbericht wurden die zentralen Erkenntnisse vorgestellt, die sich bei der Entwicklung dieses Moduls mit dem IMoMeSA-Ansatz ergaben. Dabei lag der Fokus darauf, möglichst alle Inhalte der Modellierungstechnik exemplarisch am Fallbeispiel anzuwenden, um so eine Evaluierung für jedes der erarbeiteten Konzepte zu ermöglichen. Die Evaluierung des IMoMeSA-Ansatzes umfasste schließlich die Bewertung dieser Konzepte in Bezug auf ihre Relevanz für eine industrielle Anwendung sowie eine allgemeine Bewertung des wissenschaftlich-technischen Mehrwerts und des wirtschaftlichen Nutzens, insbesondere für kleine und mittlere Unternehmen. In einem letzten Schritt wurden schließlich Möglichkeiten und Maßnahmen aufgezeigt, um die erarbeiteten Ergebnisse in das industrielle Umfeld transferieren zu können. In diesem Kontext ist es auf der einen Seite notwendig, ein entsprechendes Entwicklungswerkzeug zur Umsetzung der erarbeiteten Modellierungstechnik aufzubauen. Auf der anderen Seite muss auch die Akzeptanz für eine modellbasierte Entwicklung von Maschinen und Anlagen gesteigert werden. Dieses Ziel kann beispielsweise durch eine weiterführende Evaluierung des IMoMeSA-Ansatzes an einem realen Fallbeispiel erreicht werden.

Neben den beschriebenen Transfermaßnahmen wurden im Rahmen von IMoMeSA gemeinsam mit dem projektbegleitenden Ausschuss weitere Forschungspotenziale identifiziert, die für eine Etablierung des IMoMeSA-Ansatzes notwendig sind und somit Gegenstand zukünftiger Forschungsaktivitäten sein werden. Unter diesen ist zunächst das Themenfeld der *Modularisierung* zu nennen. Speziell der Modellierungsaufwand, der derzeit eine breite Anwendung modellbasierter Entwicklungsprinzipien verhindert, kann durch eine vereinfachte Wiederverwendung bestehender Komponenten verringert werden. Hierfür gilt es sowohl die Modellierungstechnik im Sinne eines Baukastensystems zu erweitern als auch die Workflows entsprechend anzupassen. Darüber hinaus wurde die *Ergonomie* des Ansatzes aus Sicht der unterschiedlichen Disziplinen in der Entwicklung als Optimierungspotenzial erkannt. Während der aktuelle Stand der Modellierungstechnik für Entwickler aus dem Software-Bereich bereits sehr intuitiv verständlich ist, wird die Technik von Entwicklern aus der Mechanik und der Elektrik aufgrund der dort vorherrschenden Denkweise derzeit nicht vollständig durchdrungen. Hier gilt es durch die Erweiterung des IMoMeSA-Ansatzes um disziplingerechte Modellierungselemente und Vorgehensweisen einen gemeinsamen Standard zu finden, um einen interdisziplinären Modellaufbau zu ermöglichen.

Neben den notwendigen Arbeiten an der Modellierungstechnik selbst, bedarf es zusätzlicher Maßnahmen, die speziell die langfristige Etablierung modellbasierter, interdisziplinärer Entwicklungsprinzipien ermöglichen. Dazu konnte zunächst das Themenfeld der *Einführung* des IMoMeSA-Ansatzes in Unternehmen des Maschinen- und Anlagenbaus identifiziert werden. Aufgrund der Neuartigkeit der interdisziplinären Entwicklungsmethodik bedarf es geeigneter Strategien, die eine stufenweise Einführung modellbasierter Entwicklungsprinzipien im Maschinen- und Anlagenbau ermöglichen und dabei einen fließenden Übergang von den aktuell gelebten Entwicklungsprinzipien hin zu einem modellbasierten Ansatz ermöglichen. Um dieses Ziel zu erreichen, bedarf es jedoch auch geeigneter *didaktischer* Konzepte, die es ermöglichen, die eher disziplingetriebenen Vorgehensweisen bei den Entwicklern von Maschinen und Anlagen in Frage zu stellen und den Mehrwert einer interdisziplinären Herangehensweise greifbar zu machen. Neben Workshops mit einfachen Praxisbeispielen sollen in diesem Kontext auch erste universitäre Ausbildungs-konzepte für zukünftige Ingenieursgenerationen erarbeitet werden, die dazu beitragen können, den auch im universitären Umfeld noch vorherrschenden Disziplingedanken aufzulösen und eine ganzheitliche und disziplinübergreifende Denkweise in Systemen zu fördern.

Projektbewertung

Im folgenden Abschnitt werden die Ergebnisse des Forschungsprojektes IMoMeSA hinsichtlich der im Antrag formulierten Ziele bewertet. Außerdem wird dargelegt, inwiefern die geleistete Arbeit zur Erreichung der Ziele notwendig und angemessen war. Abschließend wird der Personal- und Mitteleinsatz während der Projektlaufzeit beschrieben.

Erfüllung der Projektziele

Die im Antrag des Forschungsprojektes formulierten Ziele zum Aufbau einer modellbasierten Entwicklungsmethodik (vgl. Abschnitt 1.3) konnten vollumfänglich erreicht werden. Die erarbeitete Modellierungstechnik ermöglicht es, mechatronische Systeme von einer ersten Produktidee bis zu einem virtuellen Prototyp interdisziplinär und zielgerichtet zu entwickeln. Die einzelnen Teilaspekte, wie die Integration des Anforderungsmanagements, die Generierung von Steuerungscode, die Zusammenarbeit mit bestehenden Werkzeugketten, die Integration von Fehlermöglichkeits- und -einflussanalysen sowie die Abstimmung der einzelnen Erkenntnisse aufeinander, konnten dabei wie geplant in den ganzheitlichen IMoMeSA-Ansatz integriert werden. Die Übertragbarkeit des Ansatzes auf industrielle Problemstellungen konnte anhand des gewählten Fallbeispiels verifiziert werden. Darüber hinaus wurden im Rahmen von IMoMeSA bereits wichtige Maßnahmen identifiziert, die zu einer langfristigen Etablierung der Forschungserkenntnisse im industriellen Umfeld beitragen können (vgl. Abschnitt 4.3.3).

Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die im Rahmen der Untersuchungen der einzelnen Arbeitspakete geleistete Arbeit entsprach in vollem Umfang dem begutachteten und bewilligten Antrag und war für die Durchführung des Vorhabens notwendig und angemessen.

Personal- und Mitteleinsatz

Bei dem beantragten Forschungsvorhaben handelte es sich um ein interdisziplinäres Projekt, für dessen erfolgreiche Bearbeitung die Zusammenarbeit von Industrieunternehmen, dem Institut für Werkzeugmaschinen und Betriebswissenschaften (*iwb*) sowie dem Lehrstuhl IV: Software und Systems Engineering der Technischen Universität München erforderlich war. Für die Umsetzung der Ziele wurden insgesamt zwei wissenschaftliche Mitarbeiter betraut. Die Interdisziplinarität des Forschungsthemas sorgte dafür, dass die Arbeitspakete größtenteils von beiden Forschungsstellen bearbeitet wurden, jedoch mit unterschiedlicher Schwerpunktsetzung. Der projektbegleitende Ausschuss überwachte durch regelmäßige Treffen den Projektfortschritt. Des Weiteren wurde von dessen Mitgliedern Personal für die Analysephase sowie zur Evaluierung des erarbeiteten Fallbeispiels zur Verfügung gestellt.

Die finanziellen Mittel wurden gemäß Projektantrag eingesetzt, wobei keine Großgeräte angeschafft wurden.

Ergebnistransfer in die Wirtschaft

Im folgenden Abschnitt werden tabellarisch die einzelnen Maßnahmen beschrieben, die während des Forschungsprojektes ergriffen wurden, um die gewonnenen Erkenntnisse in das industrielle Umfeld zu transferieren.

Maßnahme	**Ziel**	**Datum/Zeitraum**
Kick-off-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Projektziele und Informationstransfer zu den Unternehmen	11.10.2012
Persönliche Interviews bei den Unternehmen des PbA	Aufnahme der Erwartungen des PbA an das Forschungsprojekt	November 2012
Veröffentlichung in WiGeP-News	Darstellung der Interviewergebnisse und des Handlungsbedarfes im Werkzeugmaschinenbau	Dezember 2012
1. Status-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Zwischenergebnisse und des weiteren Vorgehens	30.01.2013
Treffen mit Siemens PLM in München	Evaluierung des Mechatronic Concept Designer als Entwicklungstool für die Umsetzung Forschungsergebnisse	07.02.2013
Workshop mit dem PbA in Köln bei Schütte	Aufnahme von Entwicklungs-prozessen sowie typischen Fehlern und Anforderungen von Werkzeugmaschinen	24.04.2013
Treffen mit Siemens PLM in München	Weiterführende Einführung in den Mechatronic Concept Designer und Absprache über die Zusammenarbeit im Rahmen des Projektes	26.04.2013
2. Status-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Zwischenergebnisse und des weiteren Vorgehens	08.05.2013
Firmenbesuch bei Liebherr in Kempten	Konkretisierung der Projekterkenntnisse an Werkzeugmaschinen	13.06.2013
Veröffentlichung im VDW-Branchenreport	Überblick: Aktueller Stand des Projektes	Juni 2013
Veröffentlichung im *iwb*-Newsletter	Gesamtüberblick des Forschungsprojektes	Juli 2013
3. Status-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Zwischenergebnisse und des weiteren Vorgehens	01.08.2013

Maßnahme	**Ziel**	**Datum/Zeitraum**
Veröffentlichung VDI-Z	Darstellung des Sichtenkonzeptes als vorbereitender Schritt für das Entwicklungsvorgehen	September 2013
4. Status-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Zwischenergebnisse und des weiteren Vorgehens	14.11.2013
Veröffentlichung ZWF	Darstellung des modellbasierten Entwicklungsvorgehens	November 2013
5. Status-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Zwischenergebnisse und des weiteren Vorgehens	18.02.2014
Workshop bei der Firma Schütte in Köln	Evaluierung der erarbeiteten Modellierungstechnik anhand des erarbeiteten Fallbeispiels	23.04.2014
Workshop bei der Firma Klotz in Kötz	Evaluierung der erarbeiteten Modellierungstechnik anhand des erarbeiteten Fallbeispiels	28.04.2014
6. Status-Sitzung des projektbegleitenden Ausschusses	Darstellung und Diskussion der Zwischenergebnisse und des weiteren Vorgehens	21.05.2014
Veröffentlichung auf dem Automatisierungs-kongress	Darstellung der erarbeiteten Methodik zur Fehlermodellierung und des entwickelten Codegenerators	01-02.07.2014
Veröffentlichung auf der SysInt in Bremen	Darstellung der erarbeiteten Methodik zum integrativen Anforderungsmanagement	02-04.07.2014
Workshop bei der Firma Chiron in Tuttlingen	Evaluierung der erarbeiteten Methodik zur automatischen Generierung von ECAD-Dateien	14.10.2014
Abschlusstreffen des projektbegleitenden Ausschusses	Darstellung und Diskussion der Projektergmebnisse und der möglichen nächsten Schritte	31.10.2014
Workshop mit Siemens PLM in München	Diskussion hinsichtlich der Integrierbarkeit der Projektergebnisse in die Siemens-Werkzeuglandschaft	19.11.2014
Treffen des PbA bei Siemens in Erlangen	Vorstellung der Ergebnisse zur Integrierbarkeit der Projektergebnisse in die Siemens-Werkzeuglandschaft	04.12.2014
Veröffentlichung auf der ICMRE in Malaysia	Darstellung der Zusammenarbeit mit bestehenden Werkzeugketten	17-18.01.2015

Veröffentlichungen

Dieser Abschnitt listet die im Rahmen des Projektes erschienenen und bald erscheinenden Veröffentlichungen in chronologischer Reihenfolge auf. Dabei wurden sowohl Fachzeitschriften als auch Konferenzen aus den Bereichen des Maschinenbaus und der Informatik berücksichtigt. Um eine Einordnung zu erleichtern, wird jeweils eine Zusammenfassung (Abstract) des Artikels angegeben.

Herausforderungen im mechatronischen Entwicklungsprozess [62]
Diese Veröffentlichung thematisiert aktuelle Hemmnisse und Handlungsfelder bei der Entwicklung von Werkzeugmaschinen. Diese wurden über Interviews mit leitenden Entwicklungsingenieuren aufgenommen, anhand derer sowohl die aktuelle Situation in den befragten Unternehmen als auch mögliche Verbesserungspotenzial innerhalb des Entwicklungsprozesses aufgedeckt wurden.

Integrierte modellbasierte Entwicklung mechatronischer Systeme macht Fortschritte [63]
Im Rahmen dieses Beitrags werden zunächst die Projektziele des Forschungsprojektes IMoMeSA sowie die daraus abgeleitete Vorgehensweise vorgestellt. Anschließend werden die ersten Erkenntnisse innerhalb der Teilbereiche des Anforderungsmanagements und der Codegenerierung präsentiert.

Durchgängig modellbasiere Entwicklung von Werkzeugmaschinen [64]
Kürzere Innovationszyklen sowie steigende Komplexität und Individualität sind nur drei von vielen Gründen, warum ein Wandel in den Entwicklungsprozessen des Werkzeugmaschinenbaus erforderlich geworden ist. Durchgängig modellbasierte Ansätze bieten durch die Ausrichtung der Entwicklungsschritte an einem integrierten, interdisziplinären Entwicklungsmodell das Potential, die Prozesse zu vereinfachen, zu beschleunigen und das Ergebnis qualitativ zu verbessern. Im vorliegenden Artikel wird ein modellbasierter Entwicklungsprozess vorgestellt, der speziell die Entwicklung von Werkzeugmaschinen unterstützen und vereinfachen soll.

Modellbasierte Entwicklungsmethode für modulare Maschinen und Anlagen [65]
Die Modularisierung von Maschinen und Anlagen wird aufgrund einer wachsenden Kundenindividualität und Variantenvielfalt zu einem entscheidenden Faktor, um an globalisierten Märkten erfolgreich zu sein. Während disziplinspezifische Modularisierungsansätze, die beispielsweise eine Wiederverwendung mechanischer Baugruppen ermöglichen, bereits etabliert sind, mangelt es noch an einer ganzheitlichen mechatronischen Modularisierung. Diese ist allerdings erforderlich, um dem zunehmend mechatronischen Charakter von Maschinen und Anlagen gerecht zu werden. In diesem Beitrag wird eine Entwicklungsmethode vorgestellt, mit der mechatronische Module durch den Aufbau eines interdisziplinären Maschinenmodells entwickelt werden können.

A Multi-Disciplinary Modeling Technique for Requirements Management in Mechatronic Systems Engineering [12]
Machines and plants continuously increase in complexity due to higher customer expectations regarding their purpose and flexibility of use. For this reason, development processes for machines and plants need to adapt continuously to the present situation. In addition, the increasing importance of electronic and software

components in machines and plants complicates the engineering since the interaction of different disciplines leads to increasing coordination efforts and problems regarding the interfaces between these disciplines. To face these challenges, the suitable handling of arising requirements is one key factor for the success of engineering processes. Therefore, a multi-disciplinary modeling technique for requirements management is presented in this paper. After a summary of approaches for model-based development and requirements management in the context of mechatronic systems engineering, the modeling technique with its concepts, terms, tasks and workflows is illustrated. To demonstrate the feasibility of the proposed approach, the modeling technique was applied to a miniaturized packaging plant, which is currently developed at the Institute for Machine Tools and Industrial Management (*iwb*) of Technische Universität München for teaching students and benchmarking engineering methods and tools. Finally, an outlook with extensions of the concept for fault-tolerant systems and the integration of further engineering tools in order to promote a consistent engineering tool-chain is given.

Interdisziplinäre Funktionsmodellierung für die Generierung von robustem Steuerungscode für fehlertolerantes Systemverhalten [14]

Die Abstimmung zwischen Entwicklern/innen unterschiedlicher Fachdisziplinen gewinnt durch eine zunehmende Funktionsintegration und einen damit einhergehenden Komplexitätsanstieg an Bedeutung. Deswegen werden von der Wissenschaft und Industrie derzeit Ansätze erforscht, die den fachlichen Austausch von Entwicklern/innen bereits in der frühen Phase durch den Einsatz eines interdisziplinären mechatronischen Systemmodells ermöglichen. In solchen Modellen werden neben der Funktionalität einer Maschine auch erste geometrische Beziehungen sowie das grundsätzliche Steuerungsverhalten abgebildet, um so ein gemeinsames Verständnis für das zu entwickelnde System zu generieren. Um diese Erkenntnisse in der disziplinspezifischen Entwicklung aufzugreifen, ist es sinnvoll, relevante Modellinhalte automatisiert in die eingesetzten Engineering-Werkzeuge der nachfolgenden Entwicklungsphasen zu überführen. Ein Beispiel bietet hierbei die automatische Generierung von Steuerungscode aus einer zuvor modellierten Steuerungslogik.
Vor diesem Hintergrund wird im vorliegenden Beitrag eine Methodik vorgestellt, die es auf Basis eines interdisziplinären Systemmodells ermöglicht, Steuerungscode zu generieren, der neben dem Gutablauf auch auf mögliche Verhaltensfehler des mechatronischen Gesamtsystems reagiert. Dazu werden im ersten Teil des Beitrags die Grundlagen der Modellierungstechnik sowie des vorgeschlagenen Entwicklungsvorgehens vorgestellt, bevor beide um die Möglichkeiten zur Modellierung von Verhaltensfehlern und deren Behandlung erweitert werden. Anschließend wird die Methode vorgestellt, die eine automatische Transformation von Modellinhalten in ein Steuerungsprogramm ermöglicht. Dabei werden zunächst Anforderungen definiert, die zu erfüllen sind, um Codegeneratoren sinnvoll in Entwicklungsprozessen des Maschinen- und Anlagenbaus einzusetzen. Darüber hinaus werden der allgemeine Aufbau des Transformators sowie die Grundstruktur des generierten Codes dargelegt. Ein praxisnahes Beispiel einer miniaturisierten Produktionsanlage dient des Weiteren zur Evaluierung der entwickelten Methode. Abschließend wird ein Ausblick auf weiterführende Forschungsfragen dargestellt, welche insbesondere einen Transfer der Erkenntnisse auf die Bereiche der mechanischen oder elektrischen Konstruktion thematisieren.

Mit interdisziplinärer Modellierungstechnik mechatronische Systeme entwickeln [66]
In diesem Beitrag wird die im Rahmen des Forschungsprojektes IMoMeSA erarbeitete interdisziplinäre Modellierungstechnik zur Entwicklung mechatronischer Systeme vorgestellt. Einen zentralen Aspekt bildet dabei das erarbeitete Sichtenkonzept, welches es ermöglicht, einzelne Modellinhalte voneinander gekapselt darzustellen. Neben der Modellierungstechnik werden auch die darauf aufbauenden Entwicklungsaufgaben der Simulation und der Verfeinerung von modellierten Systemen betrachtet.

Modellbasierte Entwicklungsmethodik hilft mechatronische Systeme zielgerichtet zu entwickeln [67]
In dieser Veröffentlichung werden die zentralen Erkenntnisse des Forschungsprojektes IMoMeSA vorgestellt. Die einzelnen Inhalte gliedern sich dabei entsprechend den einzelnen Teilzielen des Projektes: Zunächst wird beschrieben, wie ein integratives Anforderungsmanagement und Systems Engineering im Maschinen- und Anlagenbau über einen modellbasierten Ansatz realisiert werden kann. Daraufhin werden die Möglichkeiten beschrieben, wie innerhalb der erarbeiteten Modellierungstechnik Verhaltensfehler von mechatronischen Systemen modelliert und behandelt werden können. Ein weiteres Ziel von IMoMeSA bestand in der Unterstützung Zusammenarbeit mit digitalen Werkzeugen. Im Rahmen dieses Beitrags werden die erarbeiteten Mechanismen und Modelltransformationen zur Realisierung dieser Zusammenarbeit präsentiert.

From Conception to Refinement in Mechatronic Systems Engineering [68]
The complexity of mechatronic systems increases constantly due to market requirements. Traditional engineering approaches have troubles coping with the desired functionality. A major problem is the early and continuous integration and coordination of engineering disciplines (i.e. mechanic, electric, and software). To address this situation and to enable a concurrent engineering of participating disciplines, a model-based approach to early conception and subsequent refinement of mechatronic systems is proposed in this paper. This approach allows specifying a high-level mechatronic concept within a single editor and the refinement of this concept by integrating it with established engineering tools. In particular, the paper outlines the technical and methodical integration of discipline-specific tools such that refinement steps can be tested automatically and continuously. The suitability of the approach is shown along selected examples. The paper finishes with a reflection on the current state and an outlook on future research activities.

An Integrated Model-Based Approach for Mechatronic Requirements and Systems Engineering [69]
Manufacturing systems continuously increase in complexity due to higher customer expectations regarding their purpose and flexibility of use. For this reason, development processes in this context need to be adapted continuously to the present situation. In addition, the growing meaning of electronic and software components in manufacturing systems complicates the engineering activities since the interaction of different disciplines leads to increasing coordination efforts and problems regarding the interfaces between these disciplines. To face these challenges, the suitable handling of arising requirements is one key factor for the success of engineering

processes. Therefore, a multi-disciplinary modeling technique for requirements and systems engineering is presented in this paper. After a summary of approaches for model-based development and requirements engineering in the context of mechatronic systems, the modeling technique is illustrated on a meta-model level. Subsequently, the quality assurance mechanisms are introduced, which serve the purpose of analyzing mechatronic systems in terms of syntactic and semantic correctness. To demonstrate the feasibility of the proposed approach, the modeling technique was applied to a miniaturized packaging machine, which is currently developed at the Institute for Machine Tools and Industrial Management (*iwb*) of Technische Universität München for teaching students and benchmarking engineering methods and tools. Finally, an outlook with respect to possible enhancements of the modeling technique is given.

Literaturverzeichnis

[1] Abele, E.; Reinhart, G.: *Zukunft der Produktion*. Hanser, 2011.

[2] Clark, K.; Wheelwright, S.: *Managing New Product and Process Development: Text Cases.* The Free Press, 1993.

[3] VDI/VDE Gesellschaft für Mess- und Automatisierungstechnik: *Automation 2020*. <http://www.vdi.de/fileadmin/vdi_de/redakteur_dateien/gma_dateien/AT_2020_INTERNET.pdf> - 22.01.2015.

[4] Grätz, F.M.: *Simultaneous Engineering am mechatronischen System Werkzeugmaschine*. Forschungsvereinigung Werkzeugmaschinen und Fertigungstechnik e.V. (FWF), 2005. (Abschlussbericht – Forschungsvorhaben Nr. 0823)

[5] Russwurm, S.: *Die Zukunft der Industrie.* In: Sendler, U. (Hrsg.): *Industrie 4.0 – Beherrschung der Komplexität mit SysLM.* Springer, 2013, S. 21-36.

[6] Eigner, M.: *Modellbasierte Virtuelle Produktentwicklung auf einer Plattform für System Lifecycle Management*. In: Sendler, U. (Hrsg.): *Industrie 4.0 – Beherrschung der Komplexität mit SysLM*. Springer, 2013, S. 91-110.

[7] Reinhart, G.; Wünsch, G.: *Economic application of virtual commissioning to mechatronic production systems*. Production Engineering, 1. Jg., Nr. 4, S.371 - 379

[8] VDI: *Entwicklungsmethodik für mechatronische Systeme.* Verein Deutscher Ingenieure e.V. (VDI), Beuth Verlag, 2004.

[9] Stahl, T.; Völter, M.: *Model-driven Software Development*. John Wiley & Sons Ltd, Chichester, 2006.

[10] Estefan, J.A.: *Survey of Model-Based System Engineering (MBSE) Methodologies*. INCOSE MBSE Focus Group 25, 2007.

[11] Korn, G.H.: *Informationssysteme als Mittel der Entscheidungsfindung während des Produktentstehungsprozesses*. Dissertation Technische Universität Braunschweig, Vulkan, 1996 (Schriftenreihe des IWF).

[12] Hackenberg, G.; Richter, C.; Zäh, M.F.: *A Multi-Disciplinary Modeling Technique for Requirements Management in Mechatronic Systems Engineering.* In: Proceedings of the 2nd Joint International Conference on System-integrated Intelligence: New Challenges for Product and Production Engineering, Bremen 2014.

[13] Zäh, M.F.; Lindworsky, A.: *Automatic Model Generation for Virtual Commissioning.* In: CIRP: Proceedings International Conference on Competitive Manufacturing, Paris, 2010, S. 27-32.

[14] Richter, C.; Hackenberg, G.; Zäh, M.F.: *Interdisziplinäre Funktionsmodellierung für die Generierung von robustem Steuerungscode für fehlertolerantes Systemverhalten.* In: Automation 2014 – 15. Branchentreff der Mess- und Automatisierungstechnik, VDI-Verlag, 2014.

[15] Botaschanjan, J.; Hensel, T.; Hummel, B.; Lindworsky, A.: *AutoVIBN: Automatische Generierung von Verhaltensmodellen für die qualitätsorientierte Virtuelle Inbetriebnahme.* Technical Report TUM-I1012, Technische Universität München (Abschlussbericht – AiF-Forschungsvorhaben ZN 279), 2010.

[16] Botaschanjan, J.; Hummel, B.; Lindworsky, A.: *Interdisziplinäre Funktionsmodellierung im Anlagenbau.* ZWF – Zeitschrift für wirtschaftlichen Fabrikbetrieb, Ausgabe 01-02, 2009, S. 71-75.

[17] Botaschanjan, J.; Hummel, B.; Hensel, T.; Lindworsky, A.: *Integrated Behavior Models for Factory Automation Systems.* In: Proceedings of ETFA'09, 2009.

[18] Hummel, B.; Botaschanjan, J.; Hensel, T.; Lindworsky, A., Zäh, M.F.: *Simulationsmodelle für die virtuelle Inbetriebnahme*. ATZproduktion, Ausgabe 05-06, 2009, S. 18-22.

[19] Hummel, B.: *Integrated Behavior Modeling of Space-Intensive Mechatronic Systems*. Dissertation, Technische Universität München, 2011.

[20] Hensel, T.: *Modellbasierter Entwicklungsprozess für Automatisierungs-lösungen*. Dissertation, Technische Universität München, 2011.

[21] Lindworsky, A.: *Teilautomatische Generierung von Simulationsmodellen für den entwicklungsbegleitenden Steuerungstest*. Dissertation, Technische Universität München, 2011.

[22] Blanchard, B. S., Fabrycky, W. J., Fabrycky, W. J.: *Systems engineering and analysis (Vol. 4).* Englewood Cliffs, Prentice Hall, 1990.

[23] Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual.* Pearson Higher Education, 2004.

[24] Rupp, C., Simon, M., Hocker, F.: *Requirements Engineering und Manage-ment.* HMD Praxis der Wirtschaftsinformatik, Vol. 46, Nr. 3, 2009, S. 94-103.

[25] Genest, B., Muscholl, A., Peled, D.: *Message sequence charts.* In: Lectures on Concurrency and Petri Nets, Springer, 2004, S. 537-558.

[26] Trakhtenbrot, B. A., Barzdin, Y. M.: *Finite automata.* American Elsevier Publishing Company, 1973.

[27] Hackenberg, G., Campetelli, A., Legat, C., Mund, J., Teufi, S., Vogel-Heuser, B.: *Formal Technical Process Specification and Verification for Automated Production Systems.* In: System Analysis and Modeling: Models and Reusability, Springer International Publishing, 2014, S. 287-303.

[28] Clarke, E. M., Grumberg, O., Peled, D.: *Model checking.* MIT press, 1999.

[29] Siemens PLM Software Inc*.: Mechatronics Concept Designer - Ein funktionsorientierter Ansatz für den Maschinen-und Anlagenbau.* 2010.

[30] Voelcker, H., Requicha, A.: *Constructive solid geometry.* Technical Report TM-25, University of Rochester, 1977.

[31] Hamill, P.: *Unit Test Frameworks: Tools for High-Quality Software Development.* O'Reilly Media, Inc., 2004.

[32] Duvall, P. M., Matyas, S., Glover, *A.: Continuous integration: improving software quality and reducing risk.* Pearson Education, 2007.

[33] Gosling, J. (Ed.): *The Java language specification.* Addison-Wesley Professional, 2000.

[34] Geer, D.: *Eclipse becomes the dominant Java IDE*. In: Computer, Vol. 38, Nr. 7, 2005, S. 16-18.

[35] Gamma, E., Beck, K.: *JUnit: A cook's tour*. In: Java Report, Vol. 4, Nr. 5, 1999, S. 27-38.

[36] Suykens, J. A., Vandewalle, J*.: Least squares support vector machine classifiers*. In: Neural processing letters, Vol. 9, Nr. 3, 1999, S. 293-300.

[37] Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., Feuston, B. P.: *Random forest: a classification and regression tool for compound classification and QSAR modeling*. In: Journal of chemical information and computer sciences, Vol. 43, Nr. 6, 2003, S. 1947-1958.

[38] Wang, S. C.: *Artificial neural network.* In: Interdisciplinary Computing in Java Programming, Springer US, 2003, S. 81-100.

[39] Lynch, N., Segala, R., Vaandrager, F.: *Hybrid i/o automata*. In: Information and Computation, Vol. 185, Nr. 1, 2003, S. 105-157.

[40] Fritzson, P.: *Principles of object-oriented modeling and simulation with Modelica 2.1*. John Wiley & Sons, 2010.

[41] Glassner, A. S. (Ed.).: *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[42] Barnes, M.: *COLLADA*. In: ACM SIGGRAPH 2006 Courses (SIGGRAPH '06). ACM, New York, NY, USA, 2006.

[43] John, K. H., Tiegelkamp, M.: *IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. Springer, 2010.

[44] Cagle, K., Mason, M., Watt, A., Spencer, P., Corning, M., Diamond, J.: *Professional Xsl*. Wrox Press Ltd., 2001.

[45] Hairer, E., Nørsett, S. P., Wanner, G*.: Solving ordinary differential equations I: nonstiff problems (Vol. 1)*. Springer Science & Business, 2008.

[46] de Jalón, J. G., Bayo, E.: *Kinematic and dynamic simulation of multibody systems*. In: Mechanical Engineering Series, Springer, New York, 1994.

[47] Tuinenga, P. W.: *SPICE: a guide to circuit simulation and analysis using PSpice*. Prentice Hall PTR, 1995.

[48] Bernstein, H.: *Soft-SPS für PC und IPC*. In: Grundlagen moderner SPS, VDE-Verlag, 1999.

[49] Otter, M., Blochwitz, T., Elmqvist, H., Junghanns, A., Mauss, J., Olsson, H.: *Das Functional Mockup Interface zum Austausch Dynamischer Modelle*. In: Plenary talk at the ASIM workshop, Ulm, Vol. 4, No. 5, 2010.

[50] Angermann, A.: *Matlab-Simulink-Stateflow: Grundlagen, Toolboxen, Beispiele*. Oldenbourg, 2007.

[51] Sommer, W.: *Das AutoCAD 3D Praxisbuch – Modellieren und Visualisieren , ab AutoCAD 2007.* Markt+Technik, 2007.

[52] Bernecker + Rainer Industrie Elektronik GmbH: *B&R Steuerungssysteme*. <http://www.br-automation.com/de-de/produkte/steuerungssysteme/x20-system/> - 28.01.2015.

[53] Seimert, W.: *Perfekt visualisieren und planen mit Microsoft Visio 2010.* Verlagsgruppe Hüthig, Jehle, Rehm, 2011.

[54] Stich, P.; Reinhart, G.: *Mechatronic Sketching of Manufacturing Systems using Physically based Models.* In: IEEE Symposium on Industrial Electronics and Applications (ISIESA), 2013, S. 1-6.

[55] Litto, M.: *AQUIMO – Ein Leitfaden für Maschinen- und Anlagenbauer*. Verein Deutscher Maschinen- und Anlagenbau, VDMA-Verlag, 2010.

[56] Siemens PLM Software Inc. TeamCenter: *TeamCenter.* <http://www.plm.automation.siemens.com/de_de/products/teamcenter/> - 28.01.2015

[57] EPLAN Software & Service GmbH & Co. KG: *EPLAN Engineering Configuration: Die Basis für mechatronische Konfiguration.* <http://www.eplan.de/de/loesungen/mechatronik/eplan-engineering-configuration/> - 28.01.2015

[58] DIN EN 13849: *Sicherheit von Maschinen – Sicherheitsbezogene Teile von Maschinen.* Beuth, 2008.

[59] DIN EN 62061: *Sicherheit von Maschinen – Funktionale Sicherheit sicherheitsbezogener elektrischer, elektronischer und programmierbar elektronischer Steuerungssysteme.* Beuth, 2005.

[60] Stamatis, D.H.: *Failure Mode Effect Analysis: FMEA from Theory to Execution.* Quality Press, 2003.

[61] Ericson, C.A.: *Fault tree analysis – a history*. In: Proceedings of the 17th International System Safety Conference, Orlando, 1999, S. 1-9.

[62] Zäh, M.F.; Richter, C.; Hackenberg, G.: *Herausforderungen im mechatronischen Entwicklungsprozess.* Newsletter Wissenschaftliche Gesellschaft für Produktentwicklung WiGeP, Ausgabe 1, 2013, S. 4-5.

[63] Zäh, M.F.; Richter, C.; Hackenberg, G.: *Integrierte modellbasierte Entwicklung mechatronischer Systeme macht Fortschritte*. VDW-Branchenreport, Juni 2013.

[64] Hackenberg, G.; Richter, C.; Zäh, M.F.: *Durchgängig modellbasierte Entwicklung von Werkzeugmaschinen*. VDI-Z Integrierte Produktion, Jahrg. 155, Nr. 9, 2013, S. 24-28.

[65] Richter, C.; Hackenberg, G.; Zäh, M.F.: *Modellbasierte Entwicklungsmethode für modulare Maschinen und Anlagen*. Zeitschrift für wirtschaftlichen Fabrikbetrieb (ZWF), Jahrg. 108, Nr. 11, 2013, S. 818-822.

[66] Richter, C.; Zäh, M.F.; Hackenberg, G.: *Mit interdisziplinärer Modellierungstechnik mechatronische Systeme entwickeln*. VDW-Branchenreport, Mai 2014.

[67] Richter, C.; Zäh, M.F.; Hackenberg, G.: *Modellbasierte Entwicklungsmethodik hilft mechatronische Systeme zielgerichtet zu entwickeln*. VDW-Branchenreport, Dezember 2014.

[68] Hackenberg, G.; Richter, C.; Zäh, M.F.: *From Conception to Refinement in Mechatronics Systems Engineering.* In: Proceedings of the International Conference on Mechatronics and Robotics Engineering, Kuala Lumpur, 2015.

[69] Hackenberg, G.; Richter, C.; Zäh, M.F.: *An Integrated Model-Based Approach for Mechatronic Requirements and Systems Engineering*. In: Journal of Mechatronics, 2015. (Review-Prozess läuft)