Andreas Bichler, BSc

Das Konzept des Digitalen Zwillings für integriertes Anforderungsmanagement (Requirements Engineering/Model Based Systems Engineering) am Beispiel mechatronischer Produkte

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Wirtschaftsingenieurwesen-Maschinenbau

eingereicht an der

Technischen Universität Graz

Betreuer

Ass.Prof. Dipl.-Ing. Dr.techn. **Norbert Hafner**

Dipl.-Ing. **Michael Schadler** Bsc. Bsc.

Institut für Technische Logistik

Graz, Juni 2021

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum, Unterschrift

Abstract

In terms of Industry 4.0 and advancing digitization, companies are facing new challenges. One of these challenges is in the area of product development. Due to the constantly increasing demands on a product and simultaneously shorter development times, new approaches in the development process have to be considered.
The core topic of this work is to answer two research questions. The first research question deals with the implementation of the Model Based Systems Engineering (MBSE) approach in a special purpose machinery construction company and the second research question is dedicated to automated requirements verification.
At the beginning of the thesis, the underlying theory is discussed. This includes requirements management, systems engineering, MBSE and model-based virtual product development.
For the current state of research, the literature from research projects that have already been carried out is used.
Applied research is chosen to answer the research questions. The modeling language, the method and the software are selected at the beginning. In the course of selecting the three factors, technical requirements and requirements due to the area of application are taken into account. After the selection of the three factors, the system models are created. Existing products from Fill GmbH are used as the basis for creating the system models. This ensures that all relevant information from products of this type can be mapped in the system model. The first research question is answered through the process of creating system models. The second research question is answered by using the data in the system models in combination with a simulation program via interfaces.
The result of this work are two system models of existing products and plugins that can process data from the system model via interfaces.

Kurzfassung

Im Zuge von Industrie 4.0 und der voranschreitenden Digitalisierung werden Unternehmen vor neue Herausforderungen gestellt. Eine dieser Herausforderungen befindet sich im Bereich der Entwicklung von Produkten. Durch die ständig steigenden Anforderungen an ein Produkt bei gleichzeitig kürzer werdenden Entwicklungszeiten, müssen neue Herangehensweisen im Entwicklungsprozess in Betracht gezogen werden.
Kernthema dieser Arbeit ist die Beantwortung zweier Forschungsfragen. Die erste Forschungsfrage beschäftigt sich mit der Umsetzung des MBSE Ansatzes in einem Sondermaschinenbauunternehmen und die zweite Forschungsfrage widmet sich der automatisierten Anforderungsüberprüfung.
Zu Beginn der Arbeit wird auf die zugrundeliegende Theorie eingegangen. Diese umfasst das Anforderungsmanagement, Systems Engineering, MBSE und die modellbasierte virtuelle Produktentwicklung.
Für den aktuellen Forschungsstand wird die Literatur von bereits durchgeführten Forschungsprojekten verwendet.
Zur Beantwortung der Forschungsfragen wird die angewandte Forschung gewählt. Hierbei wird zu Beginn die Auswahl der Modellierungssprache, der Methode und der Software durchgeführt. Im Zuge der Auswahl der drei Faktoren werden neben den rein technischen Anforderungen, auch Anforderungen aufgrund des Einsatzgebietes berücksichtigt. Im Anschluss daran werden die Systemmodelle erstellt. Als Grundlage für die Erstellung der Systemmodelle werden bereits bestehende Produkte der Firma Fill GmbH verwendet. Hierdurch wird sichergestellt, dass alle relevanten Informationen von Produkten dieser Art im Systemmodell abbildbar sind. Durch den Prozess des Erstellens von Systemmodellen wird die erste Forschungsfrage beantwortet. Die zweite Forschungsfrage wird beantwortet, indem die Daten in den Systemmodellen in Kombination mit einem Simulationsprogramm über Schnittstellen verwendet werden.
Das Ergebnis dieser Arbeit sind zwei Systemmodelle von bestehenden Produkten und Plugins, die über Schnittstellen Daten aus dem Systemmodell verarbeiten können.

Danksagung

An dieser Stelle möchte ich mich bei all denjenigen Personen bedanken, die mich während der Anfertigung dieser Masterarbeit unterstützt und motiviert haben.
Zuerst gebührt mein Dank Ass.Prof. Dipl.-Ing. Dr.techn. Norbert Hafner und Dipl.-Ing. Michael Schadler Bsc. Bsc., die meine Masterarbeit seitens der TU Graz betreut und begutachtet haben. Für die hilfreichen Anregungen und die konstruktive Kritik bei der Erstellung dieser Arbeit möchte ich mich herzlich bedanken.
Weiters möchte ich mich bei Dipl.Ing Alois Wiesinger Msc. und Christian Rosner bedanken, die seitens der Firma Fill diese Arbeit betreut haben. Vielen Dank an dieser Stelle für die ausgezeichnete Betreuung und die hilfreichen Inputs für die Masterarbeit.
Abschließend darf ich mich noch bei meiner Familie und meiner Freundin bedanken, die mich während der gesamten Studienzeit unterstützt haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

1 Einleitung

1.1 Umfeld und Hintergrund

Die Themen Digitaler Zwilling oder Model Based Systems Engineering MBSE stellen viele Unternehmen vor große Herausforderungen. Zu diesen Herausforderungen zählen unter anderem die Umsetzung dieser Themen parallel zum laufenden Betrieb, da unter Umständen das nötige Know-How fehlt oder die Akzeptanz für diese Themen im Unternehmen nicht ausreichend vorliegt. Des Weiteren können falsch umgesetzte Vorhaben schnell zu erheblichen Kosten ohne nennenswerten Nutzen führen. Um dieses Risiko zu verringern, bedarf es für das jeweilige Unternehmen einer genauen Analyse der Anforderungen, die mit Hilfe dieser neuen Technologien erfüllt werden sollen. Beide Themen, Digitaler Zwilling und MBSE, sind eng gekoppelt mit dem Entwicklungsprozess eines Produktes. Welche Möglichkeiten es gibt, den Entwicklungsprozess neu zu gestalten und an die neuen Anforderungen anzupassen, wird in dieser Arbeit untersucht.
Das Thema für die vorliegende Masterarbeit wurde von der Firma Fill GmbH aus Gurten vorgeschlagen. Die Firma Fill entwickelt und fertigt Sondermaschinen für die Branchen Automotive, Aerospace, Sport, Holz und Bau und beschäftigt im Jahr 2020 rund 920 MitarbeiterInnen. Neben den klassischen Sondermaschinen werden auch wiederverwendbare Module angeboten. In den überwiegenden Fällen werden diese Module in Verkettung mit weiteren Modulen oder Anlagenteilen angeboten. Beispiele dieser wiederverwendbaren Module sind Bearbeitungszentren, Gießereimaschinen, Bandsägen für die Holzbranche oder Maschinen für die Skiherstellung. Ein weiterer großer Betätigungsbereich bei Fill ist das Thema Digitalisierung und Internet of Things (IoT). Ein Produkt der Anstrengungen in diese Richtung ist beispielsweise die Softwareplattform "Fill Cybernetics". Diese Plattform verbindet alle Maschinen einer Produktionsstätte und erweitert diese um intelligente Algorithmen. Dadurch kann die Qualität und Profitabilität der Produkte gesteigert werden. Durch die ständig steigenden Anforderungen an eine Maschine und die kürzer werdenden Entwicklungszeiten, wird der zur Zeit verwendete dokumentenbasierte Ansatz in der Entwicklung zunehmend in Frage gestellt. Abhilfe soll dabei der MBSE Ansatz schaffen, der gänzlich neue Möglichkeiten in der Entwicklung und im Produktlebenszyklus schafft.
Laut Eigner belegen Statistiken, dass sich der Produktentwicklungsprozess (PEP) in den letzten Jahren ständig im Wandel befindet. Grund dafür sind sich verändernde Marktbedingungen und Anforderungen, sowie die steigende Produktkomplexität. Aus diesen Gründen entsteht die Notwendigkeit im Produktentwicklungsprozess Änderungen vorzunehmen. Die grundsätzliche Vorgehensweise dabei ist, die Tätigkeiten im Engineering , im Gegensatz zur aktuellen isolierten Betrachtung, über den gesamten Produktlebenszyklus und alle Bereiche eines Unternehmens gemeinsam zu betrachten. ([Eig17], S.5)

Um das zu ermöglichen, braucht es den MBSE Ansatz, der eine grundlegende Änderung in der Vorgehensweise zur Entwicklung von Produkten darstellt.

1.2 Beschreibung des Umfanges der Arbeit

Die Arbeit beschäftigt sich mit dem Einsatz von neuen Methoden im Produktenwicklungsprozess eines Sondermaschinenbauers. Im Umfeld eines Sondermaschinenbauers treten im Produktentwicklungsprozess bestimmte Faktoren stärker auf, als bei einem Unternehmen, welches hauptsächlich Serienprodukte entwickelt und fertigt. Beispiele dieser Faktoren sind der große Neuheitsgrad der zu entwickelnden Produkte oder die häufigen Änderungen im Laufe des Entwicklungsprozesses.
Als Ausgangspunkt für diese Arbeit wurde der MBSE Ansatz seitens der Firma Fill festgelegt. Wie dieser umgesetzt werden kann, wird im Zuge dieser Masterarbeit herausgearbeitet. Ein weiterer wichtiger Punkt dieser Arbeit ist das Anforderungsmanagement. Es soll gezeigt werden, wie Anforderungsmanagement in einem Produktentwicklungsprozess integriert werden kann. Mit *Integration* ist ein Automatismus gemeint, bei dem Teile des Anforderungsmanagements automatisiert erledigt werden können. Der MBSE Ansatz stellt die Basis für das Konzept des Digitalen Zwillings in der Produktentwicklung dar, welcher in den ersten Phasen des Produkentwicklungszyklus ein digitales Abbild des Produktes in Form eines Systemmodells ist. Des Weiteren wird durch den MBSE Ansatz das integrierte Anforderungsmanagement ermöglicht, indem die Anforderungen und die zu erfüllenden Parameter im Systemmodell in geeigneter Form gespeichert sind.

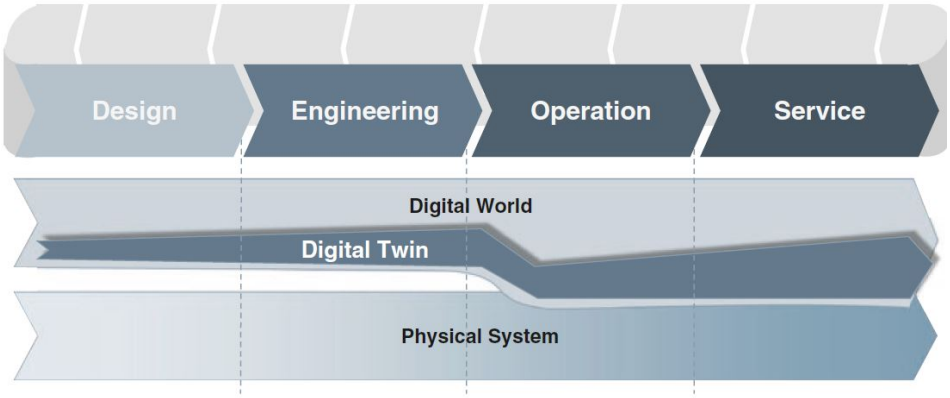


Abbildung 1.1: Entstehung eines digitalen Zwillings entlang des Produktlebenszyklus nach ([BR16], S.69)

Bezogen auf die Abbildung 1.1 wird in dieser Arbeit der Digitale Zwilling in der Designphase betrachtet. In dieser Phase werden die Anforderungen, Struktur und Funktionen eines Systems festgelegt und im Systemmodell gespeichert. Dieses Systemmodell bildet die Basis für einen Digitalen Zwilling. Im Zuge dieser Arbeit wird untersucht, wie die Umsetzung eines MBSE Ansatzes in einem Sondermaschinenbauunternehmen gelingen kann und welche Faktoren dabei zu berücksichtigen sind. Des Weiteren wird eine Möglichkeit

vorgestellt, wie mittels der Daten aus dem Systemmodell, Anforderungen automatisiert überprüft werden können. Die Berücksichtigung der Tätigkeit der Firma Fill und die dadurch entstehenden Anforderungen an den MBSE Ansatz führen dazu, dass in dieser Arbeit die wichtigsten Modellierungssprachen vergleichend gegenübergestellt werden müssen. Die grundlegenden Konzepte der SysML werden in dieser Arbeit verwendet, da diese die Basis für ein Systemmodell bilden. Des Weiteren wird die SysML auch als Referenz herangezogen, um zu überprüfen, in wie weit eine andere Sprache die Möglichkeiten der SysML abdeckt, oder eventuell in anderen Bereichen Vorteile bietet.
Die Arbeit gliedert sich in folgende Bereiche. Zu Beginn wird in Kapitel 2 der theoretische Hintergrund der betreffenden Themengebiete erklärt. Darauf folgend wird in Kapitel 3 der aktuelle Forschungsstand näher erläutert. In Kapitel 4 wird die Methodik und die praktische Umsetzung erläutert. Hierbei wird unter anderem die Vorgehensweise zur Auswahl der Sprache und des Softwaretools behandelt. Des Weiteren wird beschrieben, wie die Systemmodelle erstellt werden und in wie weit Daten aus dem Systemmodell in Kombination mit dem Anforderungsmanagement verwendet werden können. Im vorletzten Kapitel werden die Resultate und Erkenntnisse dargestellt und abschließend erfolgt in Kapitel 6 die Zusammenfassung und der Ausblick.

1.3 Forschungsfragen

Als Grundlage für die Ableitung der Forschungsfragen dient der Forschungs- und Implementierungsbedarf der Firma Fill. Dieser umfasst die Anforderungsermittlung eines Bearbeitungszentrums sowie das Erstellen eines Systemmodells in einer für die Firma Fill geeigneten Form. Weiters soll eine Möglichkeit gefunden werden, die Anforderungen, welche im Systemmodell gespeichert sind, durch simulierte Testfälle überprüfen zu können. Basierend auf diesen Aufgabenstellungen wurden zwei Forschungsfragen abgeleitet. Diese Fragen lauten:

1. Wie kann eine Umsetzung des MBSE Ansatzes in einem Sondermaschinenbauunternehmen erfolgen?

2. Wie kann eine automatisierte Überprüfung von Anforderungen durchgeführt werden?

Im Zuge der ersten Frage werden die Besonderheiten der Produkte eines Sondermaschinenbauers bei der Auswahl der nötigen Werkzeuge, Sprachen und Methoden beachtet. Im Vergleich zur Serienproduktion gibt es andere Anforderungen, die der MBSE Ansatz erfüllen muss. Beispielsweise spielt im Sondermaschinenbau die Geschwindigkeit zur Erstellung eines Modells eine erhebliche Rolle, da die Projekte oft straffen Zeitplänen unterliegen. Zur Beantwortung der ersten Frage werden unterschiedliche Softwaretools in Kombination mit Modellierungssprachen untersucht. Die zweite Frage zielt auf das Thema Anforderungsmanagement ab. Im Zuge dessen wird untersucht, wie die automatisierte Überprüfung von Anforderungen durchgeführt werden kann, welche Voraussetzungen dafür notwendig sind und wie weit eine Automatisierung möglich ist.

2 Theoretischer Hintergrund

In diesem Kapitel wird der theoretische Hintergrund der Arbeit näher erläutert. Die vorliegende Arbeit umfasst eine Vielzahl an verschiedenen Themengebieten, was dazu führt, dass dieser Teil der Arbeit sehr umfangreich ausgeführt ist.

2.1 Anforderungsmanagement

Anforderungsmanagement (AM) umfasst Maßnahmen für den richtigen Umgang mit Anforderungen. Dokumente, die in den meisten Produktentwicklungsprozessen bereits verwendet werden, sind das Lasten- und das Pflichtenheft. Im Lastenheft werden die Wünsche des Kunden in Form von Texten festgehalten. Im Pflichtenheft beschreibt der Auftragnehmer, wie er die Wünsche des Kunden umsetzen wird. Im Zuge des Anforderungsmanagements werden diese Dokumente als Quelle für Anforderungen herangezogen. Die Grundlage für die erfolgreiche Erstellung eines Systemmodells ist ein professionelles AM. Dazu gilt es mehrere Punkte zu beachten, die auf den folgenden Seiten aufgeführt sind. Mit den erstellten Anforderungen wird beschrieben, was genau gewünscht wird. Diese Wünsche werden im AM mit verschiedenen Methoden und Tätigkeiten in eine Form gebracht, um diese im weiteren Entwicklungsprozess zu verwenden.([Gra14], S.6) Anders ausgedrückt dient AM dazu, die Wünsche des Kunden herauszufinden und diese richtig umzusetzen. Für ein vertieftes Verständnis des Anforderungsmanagements sind zu Beginn ein paar Begriffserklärungen nötig.

2.1.1 Anforderungen

Würde man Anforderungen sehr allgemein beschreiben, könnte man diese auch als Kundenwünsche bezeichnen. Der Unterschied besteht allerdings darin, dass eine Anforderung kein Wunsch bleiben darf, sondern auch wirklich umgesetzt werden muss. Anders ausgedrückt beschreiben Anforderungen die Funktionalitäten, Eigenschaften und Qualitäten, die ein Produkt bekommen soll. Anforderungen sind grundsätzlich in allen Lebensbereichen zu finden, beispielsweise beim Einkaufen von Lebensmitteln oder bei der Auswahl eines neuen Autos.([Gra14], S.5) Im täglichen Leben werden Anforderungen oft nicht als solche wahrgenommen, obwohl wir ständig unbewusst damit zu tun haben. Der erste Schritt für erfolgreiches Anforderungsmanagement ist daher das Bewusstsein zu schaffen, die Wünsche des Kunden aktiv als Anforderungen zu verstehen, um so sicherzustellen, dass jeder Wunsch umgesetzt wird. Damit Anforderungsmanagement erfolgreich durchgeführt werden kann, müssen die niedergeschriebenen Anforderungen gewissen Kriterien entsprechen. Ein zentraler Punkt dabei ist, welche Attribute eine Anforderungen enthalten soll. In Summe gibt es fünf Pflicht-Attribute, die jede Anforderung besitzen soll.

Diese Attribute sind *ID, Beschreibung, Version, Status und Name.*([Gra14], S.40) Eine *ID* ist für die eindeutige Identifikation im späteren Entwicklungsprozess notwendig. Die Beschreibung enthält den eigentlichen Wunsch des Kunden. Diese muss eindeutig formuliert sein und sollte möglichst wenig Interpretationspielraum zulassen. Darüber hinaus soll die *Beschreibung* so formuliert werden, dass alle Beteiligten verstehen, was damit gemeint ist. Das Attribut *Version* ist notwendig, sobald Teile einer Anforderung geändert werden. Ist das der Fall, ermöglicht die Versionierung einen Überblick über bearbeitete und aktuelle Stände. Ein weiters Pflichtattribut ist der *Status*. Welche Stadien eine Anforderung einnehmen kann, ist je nach Art des Unternehmens oder des Produktes unterschiedlich. Typsche Stadien sind beispielsweise: Genehmigt, Freigegeben, Erfüllt etc. Das Attribut *Name* kann als Überschrift der Beschreibung verstanden werden. Dieses Attribut dient zur leichteren Zuordenbarkeit der betreffenden Anforderung. In der Abbildung 2.1 sind die Pflichtattribute grafisch dargestellt. Für Anforderungen, welche messbare oder simulierbare Größen beschreiben, kann es sehr hilfreich sein, einen Parameter als Attribut hinzuzufügen. Das vereinfacht die Verwendung der Anforderungen im weiteren Entwicklungsprozess erheblich.

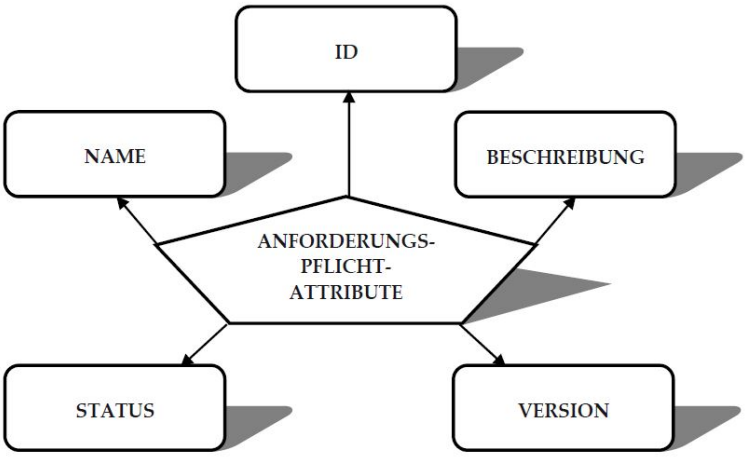


Abbildung 2.1: Notwendinge Attribute von Anforderungen nach ([Gra14], S.40)

2.1.2 Notwendigkeit von Anforderungen und Anforderungsmanagement

Anforderungen dienen dazu, dass der Kunde das Produkt bekommt, das er sich tatsächlich auch gewünscht hat.([Gra14], S.6) Ohne Anforderungen kann leicht der Fall eintreten, dass bestimmte Wünsche falsch verstanden werden, oder aber auch schlichtweg vergessen werden, da der nötige Überblick fehlt. Da Anforderungen einem gewissen Formalismus unterliegen, kann verhindert werden, dass unterschiedliche Auffassungen von ein und demselben Wunsch entstehen. Im praktischen Einsatz ist damit zu rechnen, dass der Aufwand, der zur Verschriftlichung der Anforderungen notwendig ist, kritisch gesehen wird. Da die Anforderungen die Basis aller weiteren Entwicklungsschritte bil-

den, ist es aber notwendig diesen Mehraufwand zu betreiben, um das Fundament eines erfolgreichen Entwicklungsprojektes zu legen.([Gra14], S.7) Unter AM versteht man alle Tätigkeiten, die notwendig sind, um im Entwicklungsprozess effektiv mit den festgelegten Anforderungen arbeiten zu können. Unzureichendes Anforderungsmanagement erkennt man leider oft erst dann, wenn der PEP schon sehr weit fortgeschritten ist. Wie allgemein bekannt, sind Fehler, die im späten Entwicklungsprozess auftreten, viel teurer zu beheben, als solche, die in früheren Phasen entdeckt werden. Zur Untermauerung dieser Aussagen sind in der Abbildung 2.2 die relativen Kosten der jeweiligen Phasen im Entwicklungsprozess abgebildet. Diese Abbildung zeigt das Potential von professionellem Anforderungsmanagement. Bezogen auf die in der Abbildung 2.2 verwendeten Phasen, führt richtig umgesetztes Anforderungsmanagement dazu, dass viele Fehler bereits in den Phasen *Definition* oder *Entwicklung* entdeckt, beziehungsweise vermieden werden. Dieser Umstand hat ein erhebliches Kosteneinsparungspotential, sowie eine bessere Qualiät des Produktes zur Folge.

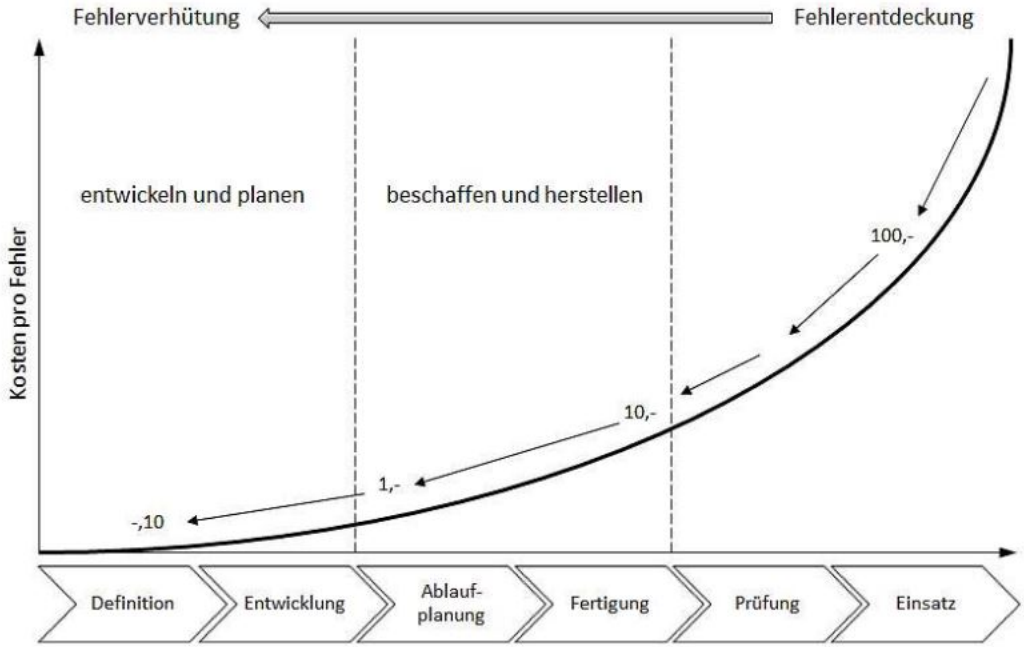


Abbildung 2.2: Zehnerregel der Fehlerkosten nach ([PS10], S.154)

2.1.3 Tätigkeiten im Anforderungsmanagement

Anforderungsmanagement stellt einen strukturierten Prozess dar, der im Idealfall dazu führt, dass keine Anforderungen vergessen werden und das finale Produkt exakt den Wünschen des Kunden entspricht. Um erfolgreiches Anforderungsmanagement betreiben zu können, gilt es vier Haupttätigkeiten zu betrachten. Diese vier Haupttätigkeiten umfassen das Ermitteln der Anforderungen, das Dokumentieren eben dieser, das Abstimmen und zu guter letzt das Verwalten der Anforderungen. Eine grafische Darstellung zum

Ablauf dieser Tätigkeiten ist in der Abbilung 2.3 zu sehen.

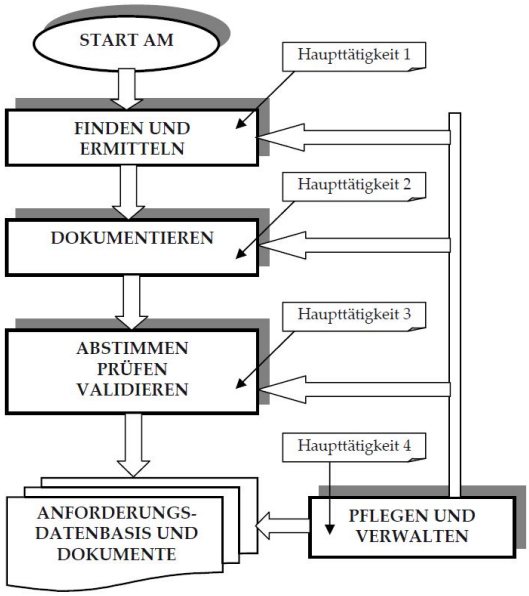


Abbildung 2.3: Häupttätigkeiten im Anforderungsmanagement nach ([Gra14], S.10)

Eine genauere Auflistung der Tätigkeiten im AM sieht folgendermaßen aus:

1. Anforderungen finden und ermitteln
2. Anforderungen dokumentieren
3. Anforderungen abstimmen und prüfen
4. Anforderungen validieren
5. Anforderungen pflegen und verwalten
6. Anforderungs-Datenbasis und Anforderungsdokumente pflegen
7. Beurteilung von Risiken der Anforderungen
8. Anforderungsmanagement mit Hilfe von Software durchführen ([Gra14], S.7)

Wie die obige Aufzählung der Tätigkeiten vermuten lässt, ist AM eine sehr stark formalisierte Tätigkeit. Dieser Formalismus führt dazu, bei den immer komplexer werdenden Produkten den Überblick behalten zu können. Um ein besseres Verständnis für Anforderungsmanagement zu entwickeln, werden die oben genannten Tätigkeiten im folgenden erläutert.

Anforderungen finden und ermitteln
Anforderungen können auf verschiedenste Art und Weise gefunden oder ermittelt werden.

Ein zentraler Punkt im Finden von Anforderungen ist die Kommunikation. Laut Grande ist es wichtig, durch Diskussion mit Stakeholdern so viele Anforderungen wie möglich herauszufinden.([Gra14], S.18)
Grundvoraussetzung dafür ist vor allem, alle wichtigen Stakeholder zu identifizieren und einzubinden. Alle relevanten Stakeholder sind für jedes Projekt über den gesamten Produktlebenszyklus (PLZ) miteinzubeziehen.
Ein weiterer Anknüpfungspunkt zum Ermitteln von Anforderungen ist laut Grande der Produktkontext. Mit Produktkontext ist die Umgebung gemeint, in der das Produkt verwendet wird. ([Gra14], S.46)
Beispielsweise gibt es länderabhängig Unterschiede in der Stromversorgung. Aus diesem Umstand heraus lassen sich bereits wichtige Anforderungen ableiten. Weitere zu betrachtende Umstände auf Grund des Produktkontextes sind beispielsweise auch das zulässige Gewicht oder die zulässigen Abmaße. Die Liste der Anforderungen aus dem Produktkontext kann sehr umfangreich werden, dient aber vor allem dazu, im laufenden Entwicklungsprozess nicht auf die äußeren Einflüsse zu vergessen. Neben den bereits genannten Möglichkeiten, Anforderungen von Stakeholdern oder aus dem Produktkontext zu erhalten, gibt es noch weitere Möglichkeiten. Eine davon ist, Anforderungen aus bestehenden Dokumenten zu extrahieren. Klassischerweise ist in diesem Fall das Lastenheft oder das Pflichtenheft gemeint, allerdings sollte man sich nicht nur darauf beschränken. Wird in einem Unternehmen Anforderungsmanagement schon längere Zeit betrieben, können unter Umständen Anforderungen zum Teil bereits aus Anforderungsdatenbanken entnommen werden. Laut Grande sind im Zuge der Anforderungsermittlung weitere Methoden und Techniken hilfreich. Anforderungen können beispielsweise in Form von Interviews oder Fragebögen ermittelt werden. Weiters können auch Kreativitätstechniken zum Einsatz kommen. Dazu zählen beispielsweise das *Brainstorming* oder der *Perspektivenwechsel*. Beim *Perspektivenwechsel* wird versucht, sich in andere Personen und ihre Rollen hineinzuversetzen. Der Wechsel der Sichweise soll dazu führen, notwendige Anforderungen zu entdecken, die ohne den *Perspektivenwechsel* verborgen geblieben wären. Eine weitere Möglichkeit Anforderungen zu ermitteln, ist es, die Abläufe durchzudenken. In diesem Zuge ist auch eine Simulation oder ein erster Prototyp sehr hilfreich. Neben all diesen Techniken und Methoden gibt es noch die Möglichkeit des Beobachtens. In diesem Fall werden durch Mitverfolgen der direkten Anwendung des Produktes Anforderungen ermittelt.([Gra14], S.49)
Nicht alle der genannten Techniken und Methoden können jederzeit bei allen Produkten angewendet werden. Beispielsweise ist für das Beobachten eines Ablaufes anhand eines Prototyps logischerweise ein vorhandener Prototyp notwendig. Darüber hinaus muss man auch unterscheiden, ob es sich bei dem zu entwickelnden Produkt um eine Neuentwicklung oder um eine Weiterentwicklung handelt. Im Falle einer Neuentwicklung fallen viele Anforderungsquellen weg, die es bei einer Weiterentwicklung gegeben hätte.
Laut Grande ist eine Einteilung sinnvoll, wie zufrieden die Stakeholder aufgrund bestimmter erfüllter Anforderuungen sind. ([Gra14], S.51) Diese Einteilung kann nach dem Kano-Modell der Kundenzufriedenheit von Kano u. a. erfolgen([Kan+84], S.39 ff). Dieses Modell schlägt die Einteilung der Anforderungen in drei Kategorien vor:

- Basisanforderungen
- Leistungsanforderungen
- Begeisterungsfaktoren

Basisanforderungen sind jene Anforderungen, die der Kunde voraussetzt, ohne diese explizit zu äußern. Für den Fall, dass Basisanforderungen fehlen, entsteht automatisch eine Unzufriedenheit beim Kunden. Leistungsanforderungen sind Anforderungen, die der Kunde explizit haben möchte und diese in der Regel auch äußert. Die Erfüllung dieser Art von Anforderungen führt zur Zufriedenheit des Kunden. Die dritte Gruppe sind die Begeisterungsfaktoren. Diese hat der Kunde nicht explizit erwähnt, allerdings entsteht durch Umsetzen solcher Anforderungen ein höherer Wert und ein besserer Eindruck des Produktes. In der Abbildung 2.4 sind die drei Typen von Anforderungen visualisiert.

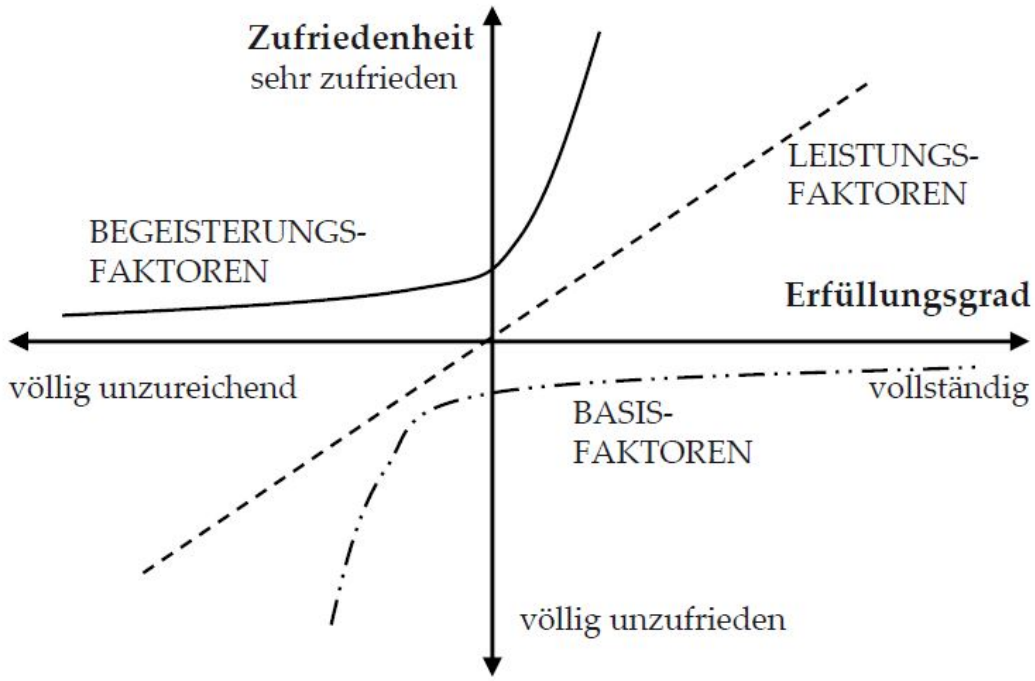


Abbildung 2.4: Kano-Modell nach ([Gra14], S.52)

Anforderungen dokumentieren
Grundsätzlich ist eine gute Anforderungsdokumentation unerlässlich, da die Dokumentation die zentrale Verwaltung der Anforderungen darstellt. Eine geeignete Dokumentation ist in der Lage, alle notwendigen Attribute einer Anforderung abzubilden und zu verwalten. In welcher Form die Dokumentation erfolgt, kann sehr unterschiedlich sein. Die einfachste Möglichkeit ist die Dokumentation in einem Textverarbeitungsprogramm oder

in einem Tabellenkalkulationsprogramm. Beispiele hierfür sind das Lastenheft oder das Pflichtenheft. Diese stellen bereits eine Art von Anforderungsdokumentation dar, auch wenn diese nicht nach den Vorgaben des Anforderungsmanagements erstellt wurden. Die Verwendung von „Standardprogrammen“ hat den Vorteil, dass die MitarbeiterInnen in der Regel schon mit den entsprechenden Programmen vertraut sind. Allerdings besitzen diese Standardprogramme keinerlei zusätzliche Funktionen, die auf die Dokumentation von Anforderungen zugeschnitten sind. Sobald es darum geht, Versionen zu verwalten, Änderungshistorien aufzurufen oder nachvollziehbare Änderungen vorzunehmen, wird es mit dieser Art von Programmen zunehmend schwieriger, den Überblick zu behalten. Für diesen Fall gibt es speziell für Anforderungsmanagement entwickelte Software, die die notwendigen Funktionen bietet.
Laut Grande gibt es fünf Qualitätskriterien für Anforderungsdokumente. Diese Kriterien lauten:

- Struktur
- Eindeutigkeit
- Vollständigkeit
- Erweiterbarkeit und Modifizierbarkeit
- Verfolgbarkeit([Gra14], S.64)

Mit einer guten Struktur des Anforderungsdokumentes steigt die Lesbarkeit und es ist möglich Inhalte schneller zu finden. Im Zuge der Eindeutigkeit sollen Anforderungen eindeutig formuliert sein und in keinem Widerspruch zu anderen Anforderungen stehen. Ist ein Anforderungsdokument vollständig, bedeutet das, dass alle relevanten Anforderungen darin dokumentiert sind und auch die dazugehörigen Versionen enthalten sind. Da ein Produktentwicklungsprozess von ständigen Änderungen duchzogen ist, stellt die Möglichkeit der Erweiterung und Modifizierung eine Notwendigkeit dar. Ein Anforderungsdokument ist nicht als starr zu betrachten, sondern muss mit ständigen Änderungen gut zurecht kommen. Als letztes Qualitätskriterium nennt Grande die Verfolgbarkeit. Diese soll Aufschluss über den Ursprung der Anforderung geben.

Um Anforderungen zu dokumentieren gibt es viele verschiedene Möglichkeiten. Welche die geeignetste Methode darstellt, muss im jeweiligen Anwendungsfall genau untersucht werden. Je nach Branche oder Produkt sind unterschiedliche Schwerpunkte zu setzen.

Anforderungen abstimmen und prüfen
Laut Grande ist das Ziel der Abstimmung die Einigung mit allen Stakeholdern bezüglich der dokumentierten Anforderungen. Diese Abstimmung soll dazu dienen, ein gemeinsames Verständnis des Produktes zu erreichen. Bei der Abstimmung der Anforderungen ist die Hauptaufgabe der Umgang mit den entstehenden Konflikten. Das Ziel ist es, alle entstandenen Konflikte, seien es Sach-, Interessens- oder Beziehungskonflikte, aufzulösen.([Gra14], S.85)

Darüber hinaus werden die Anforderungen in diesem Schritt validiert und verifiziert. Mit der Validierung wird sichergestellt, dass die entstehenden Anforderungen eine bestimmte Zweckbestimmung erfüllen. Anders ausgedrückt beantwortet die Validierung die Frage, ob das richtige Produkt gebaut wird.
Zur erfolgreichen Validierung können sechs Prinzipien angewendet werden. Diese Prinzipien lauten wie folgt:

- Die richtigen Stakeholder an der Prüfung beteiligen
- Fehlersuche und Fehlerkorrektur in zwei verschiedene Prozesse aufteilen
- Unterschiedliche Prüfungsperspektiven verwenden
- Dokumentationsform wechseln
- Entwicklungsartefakte konstruieren
- Wiederholte Prüfungen durchführen

Durch die richtige Auswahl der prüfenden Stakeholder wird sichergestellt, dass die dokumentieren Anforderungen auch aus den richtigen Perspektiven betrachtet werden. Im Falle von fehlenden oder nicht geeigneten Stakeholdern kann es sein, dass Anforderungen in deren Bereich unzureichend geprüft werden. Weiters ist eine von der Fehlersuche getrennte Fehlerkorrektur wichtig. Diese Trennung ermöglicht die Konzentration auf die Fehlersuche ohne Unterbrechungen durch die Fehlerkorrektur. Durch Wechseln der Perspektive fallen unter Umständen Fehler auf, die ohne den Wechsel unentdeckt geblieben wären. Mit dem Wechsel der Dokumentationsform meint Grande beispielsweise eine Grafik als Text oder einen Text als Grafik darzustellen. Die Änderung in der Darstellung führt dazu, dass sichergestellt werden kann, ob ein Text oder eine Grafik richtig verstanden wurde. Im Zuge des Konstruierens von Entwicklungsartefakten und der damit einhergehenden ausführlichen Beschäftigung mit den Anforderungen, können Fehler entdeckt werden, die anders nicht aufgetaucht wären. Abschließend führt Grande die wiederholte Prüfung an. Es ist davon auszugehen, dass sich im Laufe eines Entwicklungsprozesses die Anforderungen ständig ändern. Eine erneute Prüfung ist daher sinnvoll und auch notwendig.([Gra14], S.80 f.)
Nach der vorangegangenen Validierung erfolgt die Verifizierung. Im Zuge der Verifizierung ist zu überprüfen, ob das Produkt den Anforderungen entsprechend richtig gebaut wird. Diese Überprüfung erfolgt in Form eines Systemtests, eines Abnahmetests oder eines Änderungstests. Diese Tests werden je nach Situation angewendet und durchgeführt, um zu überprüfen, ob die Anforderungen korrekt umgesetzt wurden.

Anforderungen pflegen und verwalten
Wie bereits erwähnt, ändern sich Anforderungen im Laufe eines Entwicklungsprozesses. Um trotz dieser Änderungen ein gültiges Anforderungsdokument oder eine gültige Anforderungsdatenbank gewährleisten zu können, bedarf es der Pflege und der Verwaltung

von Anforderungen. Zu Beginn eines Projektes werden die Änderungen sehr umfangreich sein. Im Zuge der Entwicklung soll die Häufigkeit der Änderungen kontinuierlich abnehmen bis keine oder nur mehr sehr wenige Änderungen auftreten. Am Anfang eines Projektes ist daher der Aufwand in der Pflege und der Verwaltung aufgrund der häufigen Änderungen höher als am Ende des Projektes.

Anforderungs-Datenbasis und Anforderungsdokumente pflegen
Das Pflegen der Datenbasis beziehungsweise der Anforderungsdokumente wird ebenso als Änderungsmanagement bezeichnet. Durch erfolgreiches Änderungsmanagement ist es möglich, trotz ständiger Änderungen, eine zuverlässige und vollständige Anforderungs-Datenbasis zu behalten.
Laut Grande gibt es im Änderungsmanagement zwei wesentliche Schritte, die befolgt werden müssen, um zuverlässiges Änderungsmanagement zu betreiben. Diese zwei Schritte sind die Änderungsanfrage und im Anschluss die Änderungsbearbeitung.([Gra14], S.105) Jeder dieser Schritte kann als Ablauf dargestellt werden. In Abbildung 2.5 ist der Ablauf einer Änderungsanfrage grafisch dargestellt.

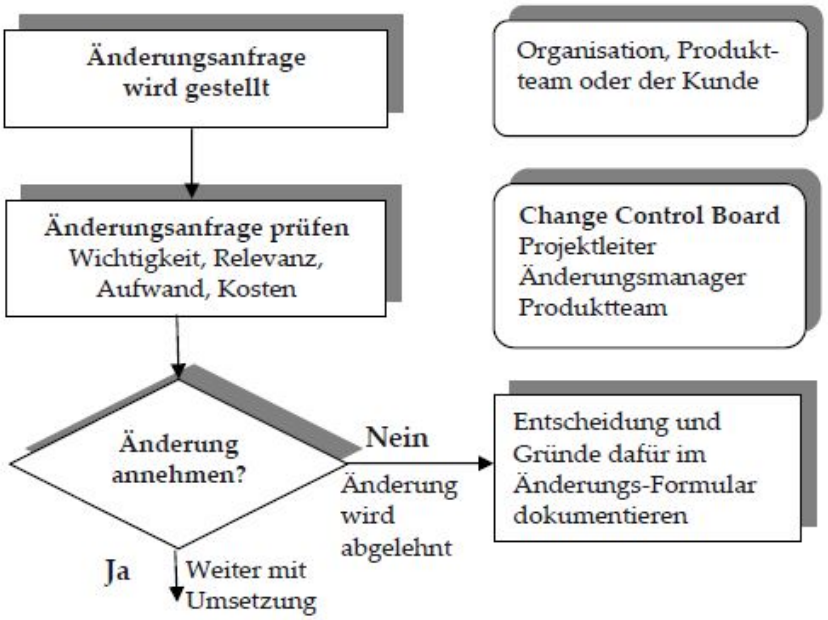


Abbildung 2.5: Ablauf einer Änderungsanfrage nach ([Gra14], S.106)

In der rechten Hälfte der Abbildung 2.5 sind die Gremien zu sehen, in der die jeweiligen Schritte umgesetzt werden. Wird eine Änderungsanfrage angenommen, erfolgt im Anschluss die Änderungsbearbeitung. In der Abbildung 2.6 ist der Ablauf im Falle einer Änderungsbearbeitung grafisch dargestellt.
Die Vorgehensweise, dass zuerst die Änderungsanfrage positiv zu absolvieren ist und im Anschluss daran die Änderungsbearbeitung erfolgt, ist unabhängig von der Speicherart der Anforderungen.

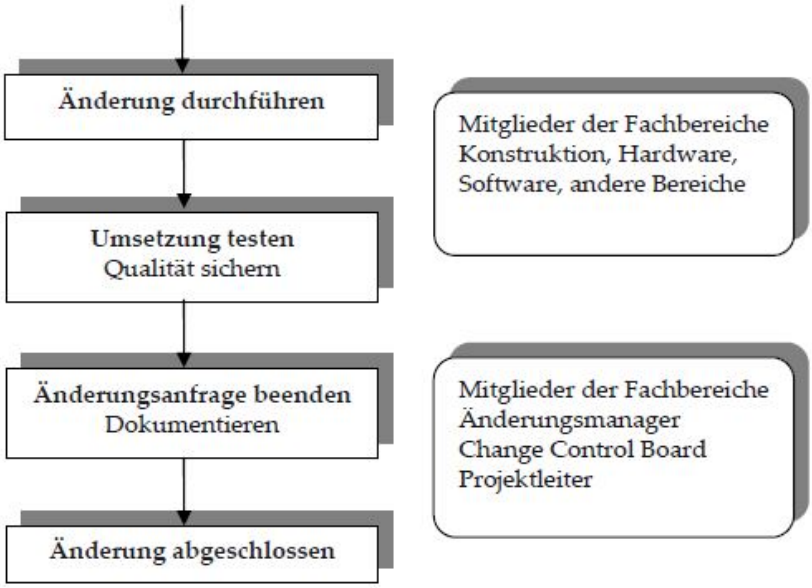


Abbildung 2.6: Ablauf einer Änderungsbearbeitung nach ([Gra14], S.108)

Beurteilung von Risiken der Anforderungen
Mit jeder umzusetzenden Anforderung ist ein gewisses Risiko verbunden. Dieses Risiko muss in der Planung berücksichtigt werden. Je mehr Erfahrung bei der Umsetzung einer bestimmten Anforderung vorliegt, desto geringer fällt das Risiko aus. Laut Grande sollen zuerst die Rahmenbedingungen, wie zum Beispiel der Zeitrahmen oder der Kostenrahmen, eruiert werden. Im Hinblick auf diese Rahmenbedingungen sollen die Risiken ermittelt werden. Daraufhin soll die Auftretenswahrscheinlichkeit und die Auswirkungen auf den Entwicklungserfolg abgeschätzt werden. ([Gra14], S.13) Mit dieser Herangehensweise wird eine Übersicht geschaffen, welche hilft, die Risiken einzuschätzen.

Anforderungsmanagement mit Hilfe von Software durchführen
Die oben genannten Schritte lassen sich grundsätzlich mit verschiedener Software durchführen. Allerdings gibt es große Leistungsunterschiede zwischen Standardsoftware wie Word oder Excel und explizit für AM entwickelte Software. Der Vorteil von Standardsoftware liegt in der Einfachheit der Bedienung und in der praktisch nicht vorhandenen Einschulungszeit der MitarbeiterInnen. Diese Vorteile werden allerdings mit zunehmendem Projektumfang unbedeutender. Vielmehr kommen bei größeren Projekten spezialisierte Software-Funktionen zum Tragen. Demzufolge sind laut Grande folgende Funktionen bei spezialisierter Software integriert:

- Ermittlung der Anforderungen

- Analyse der Anforderungen
- Spezifikation validieren
- Nachverfolgbarkeit
- Einfluss-Analyse
- Möglichkeit zur Wiederverwendung der Anforderung
- Berichte erstellen
- Konfiguration ermöglichen
- Export und Import von Anforderungen

Darüber hinaus sind im Bereich der Administration weitere Funktionen möglich:

- Anlegen von Datenbanken
- Projekt anlegen
- Erstellen von Anwendergruppen
- Verwaltung der Zugriffsrechte
- Verwalten und Erzeugen von Produktständen ([Gra14], S.29 f.)

Im Bereich der Ermittlung von Anforderungen kann eine spezialisiertes Software beispielsweise die Vergabe der Anforderungs-ID oder der Versionsnummer übernehmen. Der Anwender kann sich in so einem Fall darauf verlassen, dass die ID eindeutig ist und die Versionsnummer korrekt vergeben wurde. Bei der Anforderungsanalyse kann beispielsweise von der Software überprüft werden, ob alle notwendigen Anforderungsattribute vergeben sind. Ein anderes Beispiel ist das Filtern nach Anforderungen in bestimmten Stadien. Da sich im Entwicklungsprozess Anforderungen oft ändern, ist es wichtig zu sehen, welche Auswirkungen die Änderung einer Anforderung auf eine andere Anforderung hat. Es kann vorkommen, dass bei Änderung einer Anforderung automatisch eine andere auch geändert werden muss, da die Anforderungen voneinander abhängig sind. Des Weiteren ist im Bereich der Nachverfolgbarkeit noch anzuführen, dass es sehr hilfreich ist, stets auf die Hintergründe für eine erstellte Anforderung zurückgreifen zu können. Eine korrekt ausgeführte Nachverfolgbarkeit führt dazu, dass eine Einfluss-Analyse möglich ist. Die Einfluss-Analyse stellt bei einer Änderung die voneinander abhängigen Anforderungen dar und gibt somit Aufschluss über den Aufwand im Falle des Durchführens der Änderung. Eine Funktion zum Erstellen von Berichten mit den gewünschten Inhalten kann Zeit sparen, in der üblicherweise nur Informationen von einer Darstellungsform in eine andere gebracht werden. Übernimmt diese Arbeit eine Funktion, kann diese Zeit für andere Arbeiten verwendet werden. Der Import und der Export von Anforderungen ist nur möglich, wenn die Anforderungen in einem bestimmen Format vorliegen.

Ein solches Format ist das Requirements-Interchange-Format (RIF). Dieses Format ist ein standardisiertes XML-Format, welches es ermöglicht, Anforderungen zwischen verschiedenen Programmen auszutauschen. Das RIF hat sich im Bereich AM mittlerweile als Standard etabliert und wird von den meisten Programmen unterstützt. Im Bereich der Administration bietet professionelle Software ebenfalls viele Vorteile. Verglichen mit Standardsoftware, in denen die Anforderungen üblicherweise in Prosa anstelle einer Datenstruktur gespeichert werden, ermöglicht eine gute Datenstruktur die automatisierte Wiederverwendung der Daten. Liegen Anforderungen in einer Datenstruktur vor, können diese beispielsweise über Schnittstellen in anderen Softwaresystemen verwendet werden. Des weiteren können in professionellen Programmen Anwendergruppen definiert und auch Zugriffsrechte vergeben werden. Mit Zugriffsrechten können ungewollte oder nicht autorisierte Änderungen verhindert werden.

Anforderungsmanagement stellt ein umfangreiches Themengebiet dar, welches bei der Entwicklung von neuartigen und komplexen Produkten hilft, den Überblick zu bewahren. Des Weiteren ist es eine Herangehensweise, um alle Wünsche des Kunden zu erfüllen und gleichzeitig beziehungsweise obendrein die Qualität eines Produktes zu verbessern. Wie AM in einem Unternehmen tatsächlich umgesetzt wird und welche Programme verwendet werden, kann sehr unterschiedlich sein.

2.2 Systems Engineering

SE ist ein Ansatz, der das Systemdenken und das Erschaffen von Lösungen in einem Systemkontext fördert.
Laut Haberfellner u. a. will das Konzept Systems Engineering eine Methodik darstellen, die bei der Lösung von Problemen hilft. Hierbei ist es gleichgültig, um welche Art von Problem es sich handelt. ([Hab+18], S.9)

2.2.1 Begriffserklärung System

Für den Begriff *System* gibt es verschiedene Definitionen, abhängig davon in welchem Kontext dieser verwendet wird. Vom International Council on Systems Engineering (INCOSE) wird der Begriff folgendermaßen definiert: „A system is an arrangement of parts or elements that together exhibit behaviour or meaning that the individual constituents do not.“[INC] Eine detailliertere Definition, die auf technische Systeme zugeschnitten ist, findet sich bei Ehrlenspiel und Meerkamm: „Ein System besteht aus einer Menge von Elementen (Teilsystemen), die Eigenschaften besitzen und durch Beziehungen miteinander verknüpft sind. Ein System wird durch eine Systemgrenze von der Umgebung abgegrenzt und steht mit ihr durch Ein- und Ausgangsgrößen in Beziehung (offenes System). Die Funktion eines Systems kann durch den Unterschied, der dem Zweck entsprechenden Ein- und Ausgangsgrößen beschrieben werden[...]. Nach ihrem Verhalten können statische und dynamische Systeme unterschieden werden“([EM17], S.21). Anhand dieser Definition können beispielsweise Aktoren und Sensoren als Systemelemente gesehen werden. Zum

besseren Verständnis des Begriffes „System“ wird in der Abbildung 2.7 eine Skizze eines Systems dargestellt.

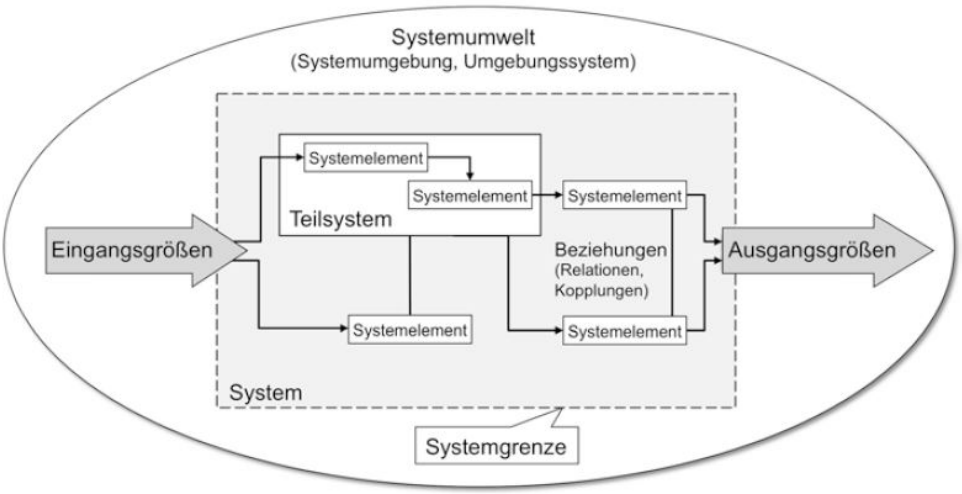


Abbildung 2.7: Bestandteile eines Systems nach ([Vaj+17], S.153)

Wie der Abbildung 2.7 zu entnehmen ist, kann ein System auch Teilsysteme enthalten. Diese Teilsysteme beinhalten ebenfalls Beziehungen und Systemelemente. Jedes System hat eine Systemgrenze, mit der das System von der Umwelt abgetrennt wird. Diese Grenze ist wichtig, um Eingangs- und Ausgangsgrößen definieren zu können.

2.2.2 Begriffserklärung Systems Engineering

Systems Engineering lässt sich am besten mit Systementwicklung ins Deutsche übersetzen. Laut Alt drückt diese Übersetzung allerdings nicht die Gesamtheit der Entwicklungsaktivitäten aus, die notwendig sind, um ein System zu entwickeln. Aus diesem Grund soll der englischsprachige Begriff Systems Engineering weiter verwendet werden. ([Alt12], S.8) Die grundlegende Vorgehensweise beschreiben Haberfellner u. a. mit dem Systems Engineering Konzept.
Das in Abbildung 2.8 dargestellte Konzept gliedert sich in die SE-Philosophie und den Problemlösungszyklus. Die SE-Philosophie beinhaltet das Systemdenken und das Vorgehensmodell. Laut Haberfellner u. a. ist der Begriff Systemdenken als Hilfsmittel zu verstehen, um Sachverhalte und Situationen zu strukturieren und in Zusammenhänge zu bringen. Das Vorgehensmodell besteht aus Grundprinzipien und Komponenten, die eine Lösung in Teilschritten ermöglichen. Der Problemlösungsprozess teilt sich auf in Systemgestaltung und Projektmanagement. Die Systemgestaltung wird wiederum in Architekturgestaltung und Konzeptgestaltung aufgeteilt. Im Bereich der Architekturgestaltung geht es um die grundlegende Struktur und das Lösungsprinzip. Bei der Konzeptgestaltung wird basierend auf der Architekturgestaltung eine konkrete Lösung herausgearbeitet. ([Hab+18], S.11)

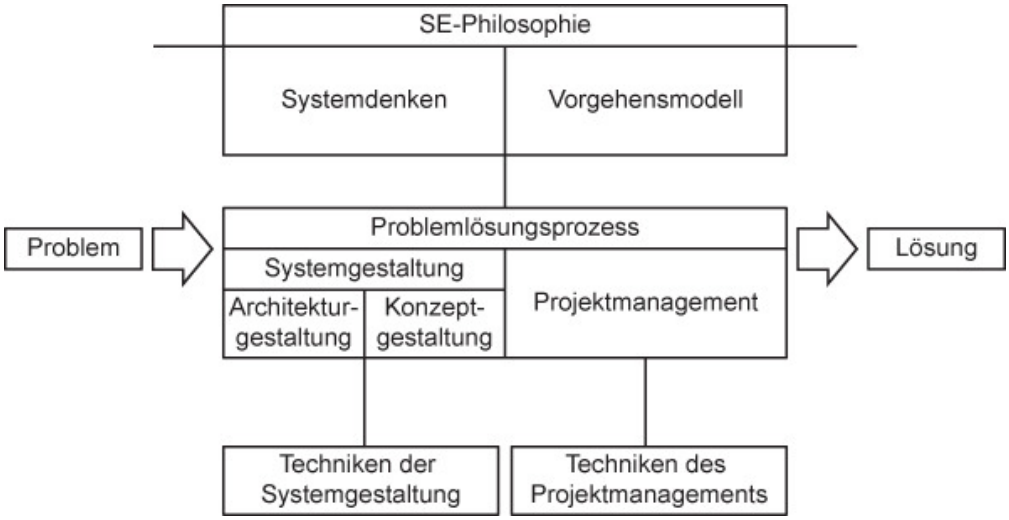


Abbildung 2.8: SE-Konzept nach ([Hab+18], S.10)

INCOSE definiert Systems Engineering folgendermaßen: „Systems Engineering is a transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods.“[INC]
Laut Alt besteht Systems Engineering aus drei wesentlichen Bausteinen. Diese sind:

- Systemarchitektur
- Systemanforderungen
- Systemverhalten

Unter Systemarchitektur sind die Struktur und die Komponenten des Systems zu verstehen. Weiters werden in der Systemarchitektur auch die Schnittstellen zwischen den Systemkomponenten, aber auch die zu den Systemgrenzen definiert. Was in der Architektur nicht enthalten ist, ist das dynamische Verhalten des Produktes. Zur Beschreibung des Verhaltens sind die verbleibenden beiden Bausteine, die Systemanforderungen und das Systemverhalten, notwendig. Mit den Systemanforderungen werden die Wünsche des Kunden niedergeschrieben. Diese Wünsche haben logischerweise Auswirkungen auf das Verhalten des Produktes. Der Baustein Systemverhalten stellt das Verhalten des Produktes in formaler Art und Weise dar. ([Alt12], S.9)
Zur Veranschaulichung der drei Bausteine des Systems Engineering ist mit Abbildung 2.9 eine grafische Darstellung dieser eingefügt.

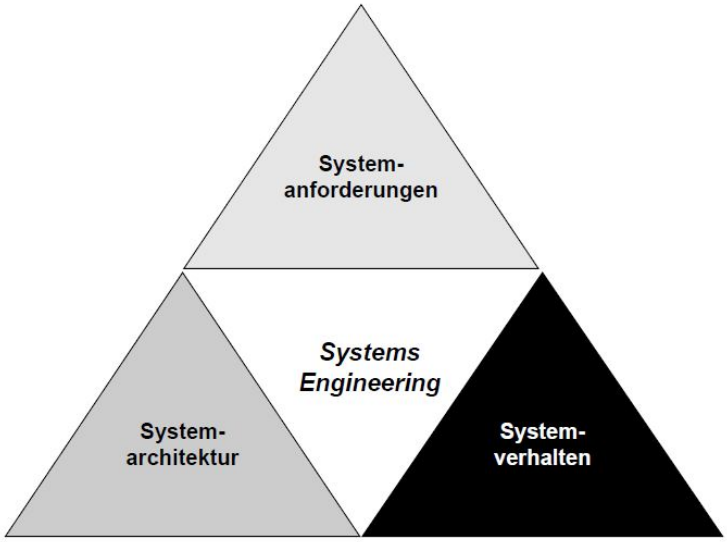


Abbildung 2.9: Bausteine des Systems Engineering nach ([Alt12], S.9)

2.3 Model Based Systems Engineering

Zur Zeit werden Projekte großteils nach dem dokumentenbasierten Ansatz abgewickelt. Dieser Ansatz hat zwar den Vorteil, dass die Informationen mit Standardwerkzeugen wie Word, Excel oder Powerpoint erstellt, beziehungsweise festgehalten werden, allerdings ist eine weitere maschinelle Verwendung dieser Informationen nicht mehr möglich. Dieser Umstand führt dazu, dass zwar die meisten Informationen in einem beliebigen Dokument niedergeschrieben sind, aber nur verwendet werden können, indem man dieses Dokument findet und die Information selber ausliest. Laut einer Studie vom Fraunhofer Institut führen die Suchzeiten nach Informationen zu großen Leistungseinbußen. Insgesamt haben an dieser Studie 913 Personen teilgenommen. Das Teilnehmerspektrum erstreckte sich quer über alle gängigen Branchen. Zweck der Studie war es herauszufinden, wie viel Zeit für das Suchen von selbst abgelegten oder bereits vorhandenen Informationen aufgewendet wird. Die für diese Tätigkeit aufgewendete Zeit erstreckt sich von 15 Minuten pro Tag bis zu über 120 Minuten pro Tag. ([Kel+09], S.54)
In der Abbildung 2.10 sind die prozentuell aufgewendeten Suchzeiten grafisch dargestellt. Auf Grund der immer umfangreicheren Produkte und der ständig steigenden Anzahl an Funktionen, die ein Produkt enthalten soll, ist die Notwendigkeit einer neuen Herangehensweise im gesamten Entwicklungsprozess immer deutlicher ersichtlich. MBSE, oder auch modellbasierte Entwicklung, ist ein Ansatz, der den Entwicklungsprozess dahingehend abändern soll, dass die zukünftig vom Kunden gewünschten Produkte zufriedenstellend entwickelt werden können.
Neben den langen Suchzeiten treten im Zuge des dokumentenbasierten Ansatzes weitere Probleme auf, die ein MBSE Ansatz lösen kann. Hierbei geht es, neben dem Vereinfachen

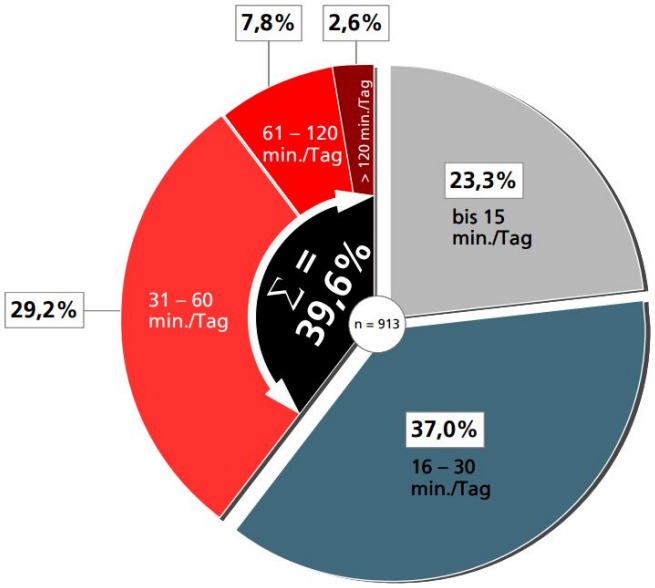


Abbildung 2.10: Suchzeiten nach Informationen bei selbst abgelegten oder bereits vorhandenen Informationen nach ([Kel+09], S.54)

der Suche nach Informationen, auch um die weitere Verwendung von bereits erstellten Informationen.

2.3.1 Begriffserklärung Modell

Grundsätzlich ist ein Modell immer eine abstrakte Darstellung der Realität. Der Begriff „Modell“ wird in vielen Bereichen verwendet und hat daher je nach Verwendungsgebiet eine unterschiedliche Bedeutung. Im Zuge des Model Based Systems Engineering wird die Darstellung des Produktes in einer bestimmten Art und Weise als Modell bezeichnet. Laut Alt gibt es bei der Definition des Begriffes Modell noch eine weitere Einschränkung. Diese besagt, dass im Zuge von modellbasierter Entwicklung erst von einem Modell gesprochen werden kann, sobald das Modell eine gewisse formale Form einhält. Diese Form soll es erlauben, dass aus dem Modell automatisiert Informationen abgeleitet werden können. Anders ausgedrückt muss es das Modell ermöglichen, dass die Modelldaten von Rechnern verarbeitet werden können. ([Alt12], S.20)
Im Umkehrschluss ist daher festzuhalten, dass Daten die in Dokumentenform (Word, Excel, Powerpoint usw.) ohne Datenstruktur vorliegen und somit nicht automatisiert von Rechnern verarbeitbar sind, nicht als Modelle im Sinne des modellbasierten Ansatzes anzusehen sind. Erst, wenn die Daten in einer bestimmten Datenstruktur festgelegt sind und diese die automatisierte Verarbeitung durch Rechner zulässt, kann von einem Modell im Sinne der modellbasierten Systementwicklung gesprochen werden.

2.3.2 Zusammenhang und Unterschied zwischen Systems Engineering und MBSE

Systems Engineering legt den Grundstein für die Entwicklung von Systemen. Laut Friedenthal et al. ist der Unterschied zwischen Systems Engineering und Model Based Systems Engineering folgendermaßen zu beschreiben. Im Systems Engineering wird vorrangig der dokumentenbasierte Ansatz verfolgt. Anders als beim modellbasierten Ansatz liegt den Daten im ursprünglichen Systems Engineering kein Datenmodell zugrunde. Es werden die Informationen, die während des Entwicklungsprozesses anfallen, in dokumentenbasierter Form festgehalten und weiter verarbeitet. ([FMS08], Kap 2.1) Demzufolge ist der MBSE Ansatz als eine Weiterentwicklung des Systems Engineering Ansatzes zu verstehen. Die grundsätzliche Vorgehensweise weist viele Parallelen auf, allerdings stellt die Art wie Informationen gespeichert, verwaltet und verarbeitet werden einen wesentlichen Unterschied dar. Die Idee von MBSE ist es, den dokumentenbasierten Ansatz durch den modellbasierten Ansatz abzulösen, da ersterer sehr schnell an die Grenzen seiner Belastbarkeit kommt. Die Überschreitung dieser Grenzen äußern sich in folgenden Punkten:

- Lange Suchzeit für vorhandene Informationen
- Nur Änderung der Informationsdarstellung, aber keine neuen Informationen (z.B. Berichte)
- Vergessene Anforderungen
- Keine Möglichkeit der maschinellen Verarbeitung der Informationen
- Keine Single Source of Truth [1]
- Verschiedene Versionen vorhanden, wobei nicht klar ist, welche die aktuellste ist
- Vorrangig Prosa Texte, die manuell nach den gewünschten Informationen durchsucht werden müssen

Grundsätzlich werden im modellbasierten Ansatz die Informationen an einer zentralen Stelle verwaltet und gespeichert. Diese zentrale Stelle nennt sich das Datenmodell. Diesem Datenmodell liegt eine Datenstruktur zugrunde, welche die Möglichkeiten schafft, die bei einem dokumentenbasierten Ansatz unmöglich zu erreichen sind. Wichtig ist, dass von Beginn des Projektes an alle anfallenden Informationen in das Datenmodell eingepflegt werden, um im Laufe des Projektes schnell und gezielt darauf zugreifen zu können. In Abbildung 2.11 sind der dokumentenbasierte und der modellbasierte Ansatz grafisch gegenübergestellt.
In der linken Hälfte der Abbildung 2.11 ist der dokumentenbasierte Ansatz dargestellt. Bei diesem Ansatz werden die Informationen untereinander in verschiedenen Formaten ausgetauscht. Diese Vorgehensweise kann beispielsweise schnell zu mehreren Versionen

[1] Beschreibt einen Datenbestand, der den Anspruch hat, korrekt zu sein und auf den man sich verlassen kann.

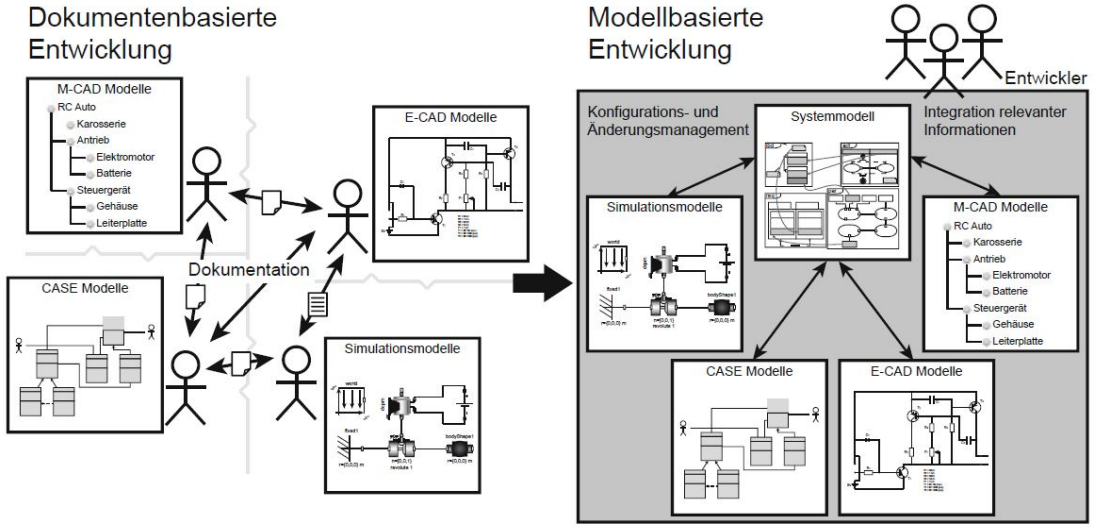


Abbildung 2.11: Grafische Darstellung des Unterschiedes zwischen dem dokumentenbasierten Ansatz und dem modellbasierten Ansatz nach ([Zaf14], S.82)

führen, von denen nicht bekannt ist, welche die gerade aktuelle ist. Ein weiteres Problem dieses Ansatzes ist, dass bei einer großen Anzahl an Entwicklern die gleiche Arbeit oft mehrmals erledigt wird, da nicht eindeutig ersichtlich ist, welche Arbeiten schon erledigt wurden. Die Wiederverwendbarkeit von Informationen in Dokumentenform ist im Vergleich zum modellbasierten Ansatz de facto nicht möglich. In der rechten Hälfte der Abbildung 2.11 ist der modellbasierte Ansatz dargestellt. Bei diesem Ansatz werden alle Informationen zentral über das Systemmodell verwaltet. Die zu verwaltenden Informationen werden in einer festgelegten Datenstruktur gespeichert. Dies ermöglicht im weiteren Verlauf des Projektes zum einen das Schnelle finden von Informationen und zum anderen das maschinelle Auslesen der Informationen. Um die Vorteile eines Systemmodells nutzen zu können, ist es notwendig, dieses in einer geeigneten Art und Weise zu erstellen.

2.3.3 Voraussetzungen für das Erstellen eines Systemmodells

Für das Erstellen eines Systemmodells müssen im Vorfeld Rahmenbedingungen geschaffen werden. Grundsätzlich müssen drei Faktoren gegeben sein, um ein Systemmodell erstellen zu können:

- Methode
- Sprache
- Werkzeug

In Abbildung 2.12 werden diese drei Faktoren zur Verdeutlichung grafisch dargestellt.

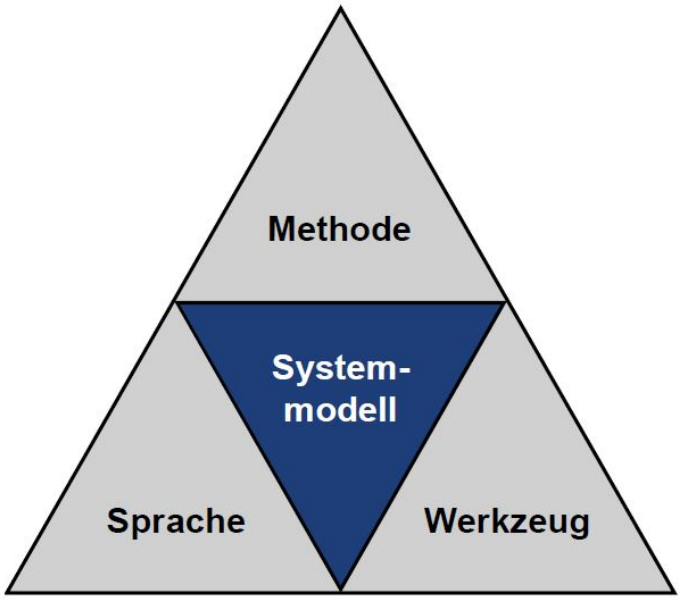


Abbildung 2.12: Grafische Darstellung des Zusammenspiels der drei Faktoren nach ([Kai13], S.26)

Jeder dieser drei Faktoren gehört gleichermaßen berücksichtigt, ansonsten ist es nicht möglich ein Systemmodell zu erstellen. Zur Festlegung der Faktoren bedarf es jeweils eines Auswahlprozesses, der je nach Anwendungsgebiet und Branche unterschiedliche Schwerpunkte setzt und somit auch unterschiedliche Ergebnisse liefert. Es ist daher zwingend erforderlich, genau zu analysieren, welche Methode, Sprache oder Werkzeug für den untersuchten Anwendungsfall benötigt werden. In der Regel werden solche Entscheidungen nicht projektabhängig, sondern unternehmens-, beziehungsweise branchenabhängig getroffen. Da sich die MitarbeiterInnen, die das Systemmodell erstellen werden, mit der Methode, der Sprache und dem Werkzeug in teilweise aufwendiger Art und Weise vertraut machen müssen, werden diese Entscheidungen für lange Zeit getroffen. Aus diesem Grund ist es sinnvoll, diesen grundlegenden Entscheidungen genug Zeit zu geben, um für das Unternehmen das beste Ergebnis zu erzielen.

2.3.4 Sprachen zum Erstellen eines Systemmodells

Es gibt verschiedene Sprachen, die das Erstellen eines Systemmodells ermöglichen. Allerdings ist nicht jede Sprache für jeden Anwendungszweck geeignet. Die wohl bekannteste Sprache zur Modellierung ist die Systems Modeling Language (SysML). Weitere Sprachen sind:

- CONSENS
- MechatronicUML

- ARCADIA Language

Bei „CONceptual desing Specification technique for the Engineering of complex Systems (CONSENS)“, „Architecture Analysis & Design Integrated Approach (ARCADIA)“ und MechatronicUML sind Sprache und Methode eng miteinander verknüpft. Aus diesem Grund besitzen Sprache und Methode in diesen Fällen dieselbe Bezeichnung. Laut Voirin werden bei ARCADIA nicht die Anforderungen als Haupttreiber der Entwicklung gesehen, sondern die funktionale Analyse ([Voi13], S.140). Aus diesem Grund wird ARCADIA in dieser Arbeit nicht näher betrachtet. Die MechatonicUML hat ihren Schwerpunkt in der Modellierung von Softwarearchitekturen. Da bei den Produkten der Firma Fill der mechanische Anteil sehr hoch ist, wird diese Sprache für diese Arbeit nicht in Betracht gezogen.
In diesem Abschnitt werden die Sprache SysML und CONSENS genauer erklärt.
Die SysML leitet sich aus der Softwaremodellierungssprache Unified Modeling Language (UML) ab. Die Ähnlichkeit dieser beiden Sprachen lässt sich anhand der verwendeten Diagramme und der Profile zeigen. Der grundlegende Unterschied ist der Verwendungsbereich. Im Gegensatz zur UML wurde die SysML entwickelt, um physische Produkte zu modellieren. Typischerweise handelt es sich bei den angesprochenen Produkten um mechatronische Produkte. Der Unterschied zwischen UML und SysML besteht daher hauptsächlich darin, dass SysML die Möglichkeit bietet, neben der Modellierung von softwarespezifischen Themenbereichen auch elektronische oder mechanische Themenbereiche zu modellieren. Um dies zu ermöglichen, wurden die Elemente und Diagramme dahingehend angepasst. In Abbildung 2.13 sind die Diagrammtypen, die in der SysML verwendet werden, ersichtlich.

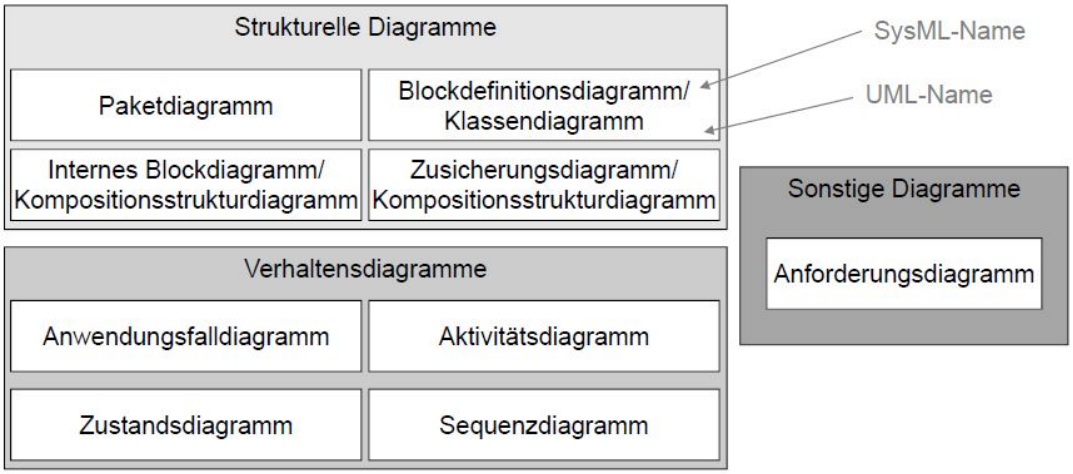


Abbildung 2.13: Diagrammtypen der SysML nach ([Alt12], S.40)

Jedes dieser Diagramme ermöglicht eine andere Sicht auf das zugrunde liegende Modell. Was der Begriff „Sicht“ bedeutet soll die Abbildung 2.14 verdeutlichen. Zum vollständigen

Verständnis wie ein Systemmodell aufgebaut ist, ist es wichtig zu beachten, was das eigentliche Modell ist, und was lediglich eine Sicht auf das Modell darstellt.

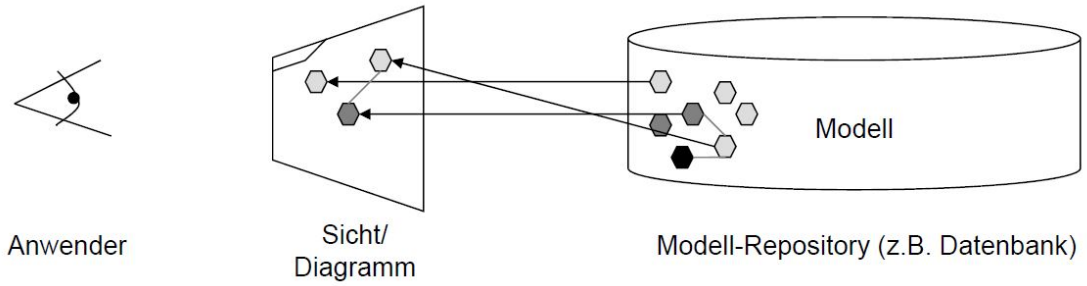


Abbildung 2.14: Unterscheidung von Sicht und Modell nach ([Alt12], S.39)

Der Anwender sieht lediglich die Darstellung des Modells in einem bestimmten Diagrammtyp. Das zugrunde liegende Modell kann beispielsweise in einer Datenbank gespeichert sein. Die für die Darstellung des Modells verwendbaren Diagrammtypen lassen je nach Typ eine unterschiedliche Sicht auf das Systemmodell zu. Beispielsweise werden durch das *Blockdefinitionsdiagramm* die Blöcke des zu modellierenden Produktes definiert. Im *Anforderungsdiagramm* werden die Anforderungen grafisch dargestellt und mit der Gruppe der *Verhaltensdiagramme* wird das Verhalten des Produktes dargestellt. Jedes dieser Diagramme wird durch SysML-Elemente aufgebaut beziehungsweise dargestellt. Bezüglich Aufbau des Diagramms beziehungsweise des Modells gilt immer das Konzept von Definition und Instanzierung. Je nach Diagrammtyp gibt es unterschiedliche Elemente, die zur Darstellung dienen. Neben den Standardelementen, die die SysML zur Verfügung stellt, besteht auch noch die Möglichkeit Profile zu erstellen. Diese Profile dienen dazu, bestehende SysML-Elemente zu erweitern, um so besser die eigenen Bedürfnisse erfüllen zu können. Zum besseren Verständnis werden die vorhandenen Diagrammtypen der SysML in den nächsten Zeilen genauer beschreiben.

Konzepte der Objektorientierung

Um mit der SysML ein Modell erstellen zu können, ist es unerlässlich, das Konzept der Objektorientierung zu verstehen und anzuwenden. In der Objektorientierung spielen Klassen und Objekte eine zentrale Rolle. Eine Klasse wird oft als Bauplan gleichartiger Objekte gesehen. Was dieses Objekt darstellt, ist vorerst nicht relevant. Beispiele für mögliche Klassen sind:

- Mitarbeiter
- Student
- Auto
- Sensor
- Aktor

Damit Objekte einer Klasse zugeordnet werden können, müssen diese die gleichen Attribute besitzen. Am Beispiel der Klasse *Student* sind folgende Attribute gleich:

- Name
- Vorname
- Matrikel-Nr.
- Studienfach
- Fachsemester ([Alt12], S.33)

Die angeführten Attribute sind nicht vollständig und können noch beliebig erweitert werden. Sobald eine Klasse definiert ist, können beliebig viele Instanzen davon erstellt werden. Ein Beispiel von Instanzen der Klasse *Student* ist in der Abbildung 2.15 zu sehen. Ausgehend von der Abbildung 2.15 stellt *student1* eine Instanz der Klasse *Student* dar.

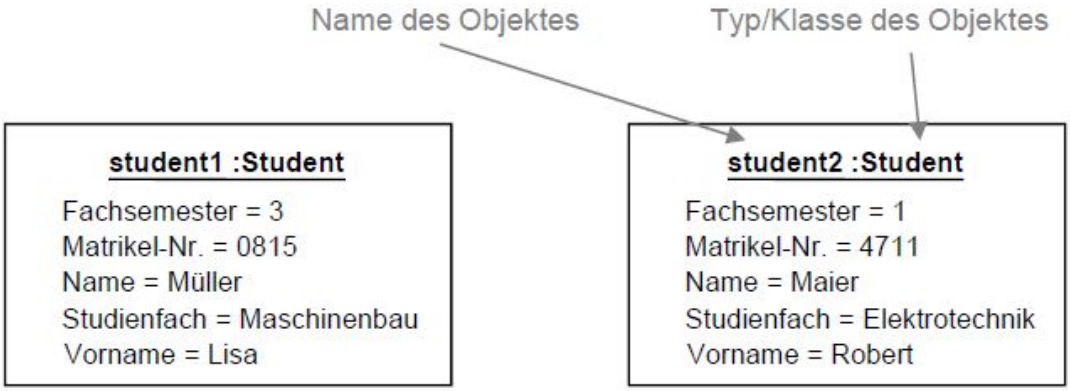


Abbildung 2.15: Instanzen der Klasse *Student* nach ([Alt12], S.34)

Die Attribute der Instanzen, welche auch als Objekte bezeichnet werden können, haben jeweils unterschiedliche Werte. Durch diese Vorgehensweise ist es möglich, verschiedene Objekte aus demselben Grundgerüst zu erzeugen. Bei der Erstellung von Systemmodellen ist diese Vorgehensweise oft in Verwendung. Neben den Klassen und Objekten gibt es auch noch das Konzept der Vererbung. Beispielsweise kann die Klasse *Student* die Attribute *Name* und *Vorname* von einer Klasse *Person* erben, da ein *Student* immer eine *Person* sein muss. Im Bezug auf die obige Aufzählung von möglichen Klassen könnte auch die Klasse *Mitarbeiter* von der Klasse *Person* erben. Die Abbildung 2.16 stellt das Konzept der Vererbung in grafischer Form dar. Bei der Entwicklung des SysML-Standards hat man sich laut Alt überlegt, die Konzepte zwar zu übernehmen, allerdings nicht die Begriffe. Diese werden von vielen Leuten sehr stark mit der objektorientierten Softwareentwicklung in Verbindung gebracht. Aus diesem Grund werden die Klassen als Blöcke bezeichnet und die Instanzen der Blöcke als Block-Properties. Um das zu verdeutlichen, werden in der Abbildung 2.17 die beiden Instanzen der Klasse Student als SysML Elemente dargestellt.

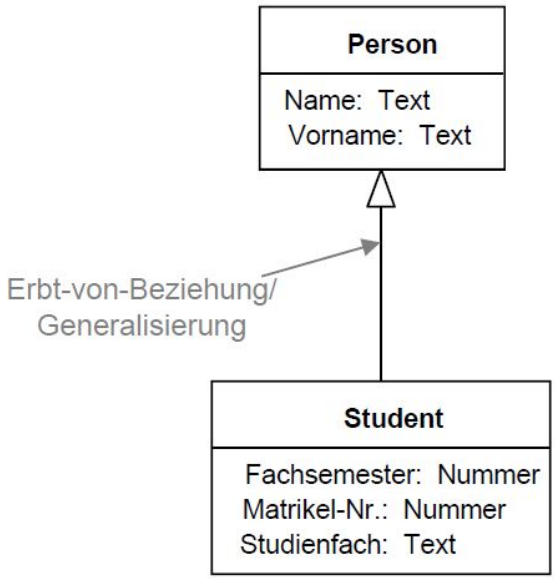


Abbildung 2.16: Das Konzept der Vererbung nach ([Alt12], S.35)

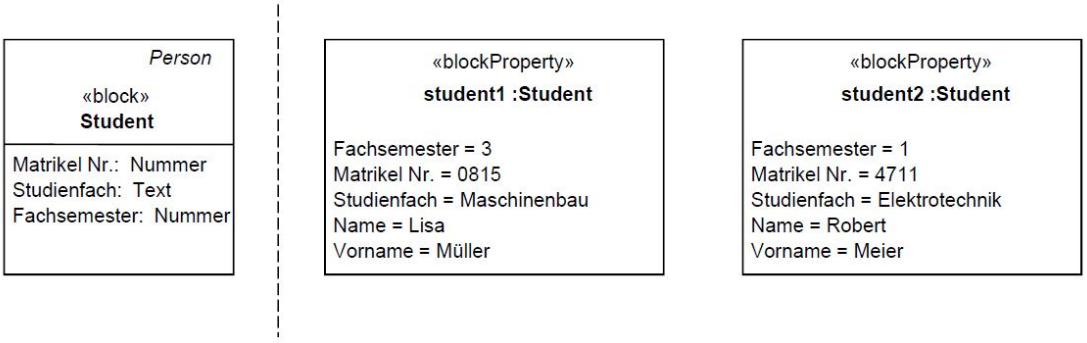


Abbildung 2.17: Instanzen der Klasse *Student* als SysML Elemente nach ([Alt12], S.37)

Die eben erklärten und grafisch dargestellen Konzepte finden in den folgenden Diagrammtypen immer wieder Anwendung.

Blockdefinitionsdiagramm
Das zum Blockdefinitionsdiagramm äquivalente Diagramm in der UML ist das Klassendiagramm. Das bedeutet, das Blockdefinitionsdiagramm wird dafür verwendet, die Blöcke (äquivalent zu Klassen in der Softwaretechnik) die zur Erstellung eines Produktes notwendig sind, zu definieren. Laut Alt bilden die Blöcke eine gemeinsame Grundlage, die zur Modellierung verwendet werden kann. Neben der Definition von Blöcken wird das Blockdefinitionsdiagramm auch verwendet, um Schnittstellen zwischen den Systemkomponenten zu spezifizieren. In der Abbildung 2.18 ist ein einfaches Beispiel eines Blockdefinitionsdiagrammes grafisch dargestellt. Darin sind Teile des Konzepts der Objektorientierung zu erkennen.

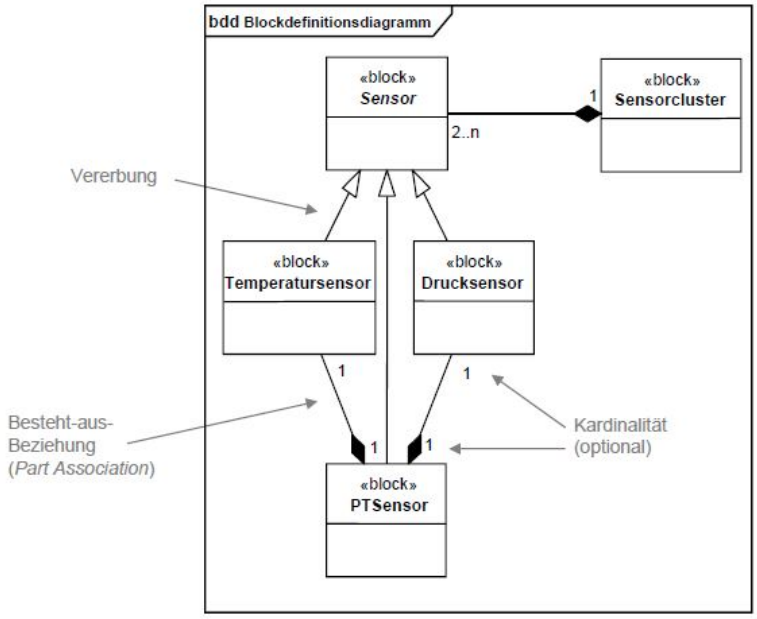


Abbildung 2.18: Beispiel eines Blockdefinitionsdiagrammes nach ([Alt12], S.43)

Nachdem die notwendigen Blöcke für das Produkt definiert sind, werden Instanzen davon beispielsweise im internen Blockdiagramm verwendet.

Internes Blockdiagramm
Das interne Blockdiagramm stellt die statische Struktur eines Systems dar. Genauer gesagt wird mit diesem Diagrammtyp die Architektur eines Systems dargestellt. Laut Alt ist dieser Diagrammtyp der am häufigsten verwendete, da die Architekturarbeit den Großteil des Systems Engineering ausmacht. Zur Modellierung der Systemarchitektur werden praktisch nur drei Modellierungselemente verwendet.

- Instanzen der Blöcke, diese stellen die Systemkomponenten dar
- Schnittstellen, diese werden als Flow Ports ausgeführt
- Verbindung zwischen den Schnittstellen, diese werden als Item-Flow Konnektoren ausgeführt

Neben diesen drei Elementen gibt es noch weitere, diese werden allerdings in der Praxis kaum eingesetzt. ([Alt12], S.46)
Das Erstellen dieser Art von Diagrammen ist, verglichen mit anderen Diagrammtypen, einfach, ebenso wie das Lesen. Beides ist mit relativ wenig Erfahrung auf Grund der geringen Anzahl an verwendeten Elementen durchführbar. Allerdings ist im Vorhinein ein Blockdefinitionsdiagramm zu erstellen, welches deutlich mehr Kenntnisse benötigt. In der Abbildung 2.19 ist ein Beispiel eines internen Blockdiagrammes dargestellt. In dieser Darstellung finden alle drei gängigen Elemente Anwendung.

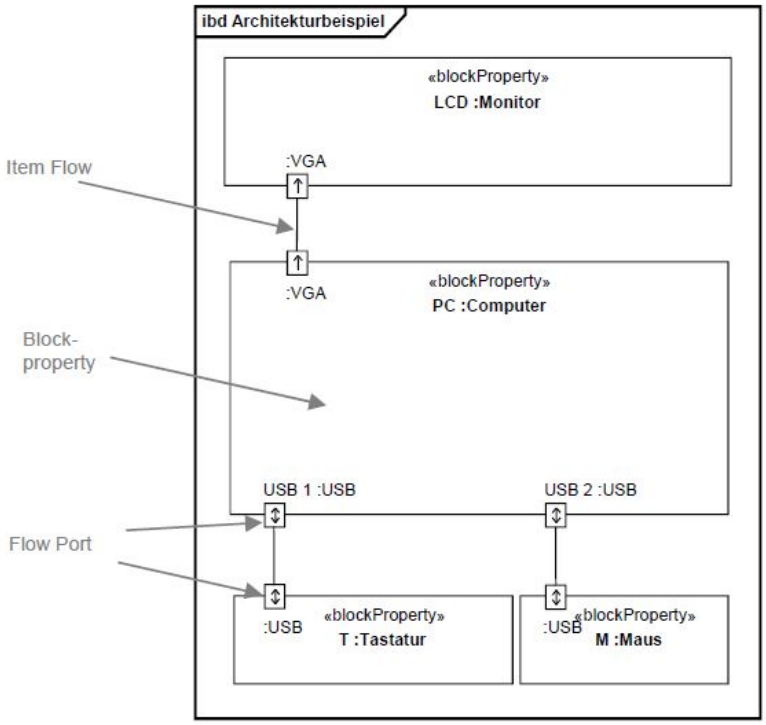


Abbildung 2.19: Beispiel eines internen Blockdiagrammes nach ([Alt12], S.46)

Wie in Abbildung 2.13 ersichtlich, gehören das Blockdefinitionsdiagramm und das interne Blockdiagramm zur Gruppe der strukturellen Diagramme. Mit diesen Diagrammen kann die für Systems Engineering erforderliche Systemarchitektur dargestellt werden. Ein weiteres Diagramm dieser Gruppe ist das Paketdiagramm.

Paketdiagramm
Das Paketdiagramm hat in der UML dieselbe Aufgabe wie in der SysML. Laut Friedenthal, Moore und Steiner stellt ein Paketdiagramm die Organisation eines Modells dar. Anders ausgedrückt enthält ein Paketdiagramm die Modellelemente in Form von Paketen. ([FMS08], Kap.3) Diese Darstellung ermöglicht eine Übersicht über umfangreiche und komplexe Modelle. Welche Modellelemente zu einem Paket zusammengefasst werden, steht dem Ersteller des Diagrammes frei. Es ist aber darauf zu achten, dass nur Modellelemente in ein Paket zusammengefügt werden, welche im Sinne des Systems zusammengehören.

Zusicherungsdiagramm
Der vierte Diagrammtyp der Gruppe der strukturellen Diagramme ist das Zusicherungsdiagramm. Das Zusicherungsdiagramm ist im Englischen auch unter dem Namen *Parametric Diagramm* bekannt. Dieses Diagramm dient dazu mathematische Gleichungen mit Parametern zu modellieren. Im Vorhinein müssen die sogenannten *Constraint Blocks* in einem Blockdefinitionsdiagramm festgelegt werden. Die *Constraint Blocks* definieren die mathematische Operation der Parameter, beispielsweise eine Addition oder eine Subtraktion. Auch im Zuge dieses Diagrammtypes ist die eingangs erwähnte Objektorientierung wichtig. Wie bereits erwähnt, werden die *Constraint Blocks* definiert und im Anschluss werden Objekte davon erstellt. In Abbildung 2.20 ist die Definition von *Constraint Blocks* grafisch dargestellt.

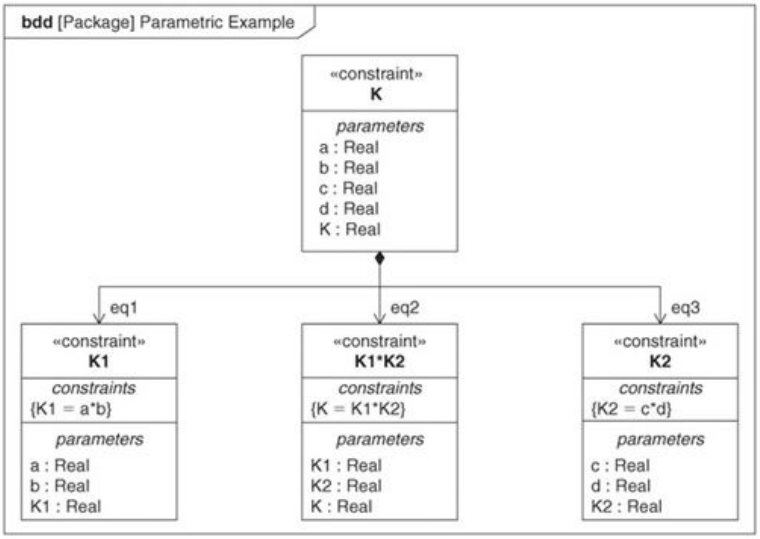


Abbildung 2.20: Definition der *Constraint Blocks* nach ([FMS08], Kap.7)

Mit der Abbildung 2.20 werden mehrere bereits erklärte Elemente des SysML dargestellt. Zunächst wird die Definition der *Constraint Blocks* veranschaulicht. Wie schon erwähnt, werden Definitionen in der SysML in Blockdefinitionsdiagrammen durchgeführt. Dass es sich in der Abbildung 2.20 um ein Blockdefinitionsdiagramm handelt, ist an den drei

Buchstaben “bdd“ in der linken oberen Ecke der Abbildung zu erkennen. Rechts neben diesen Buchstaben steht in eckigen Klammern das Wort *Package*. Das gibt Aufschluss darüber, dass diese *Constraint Blocks* zu einem *Package* zusammengeführt sind. Nach erfolgreicher Definition der *Constraint Blocks* werden in weiterer Folge Objekte davon erstellt, damit diese in einem Zusicherungsdiagramm zum Modellieren einer mathematischen Gleichung verwendet werden können.

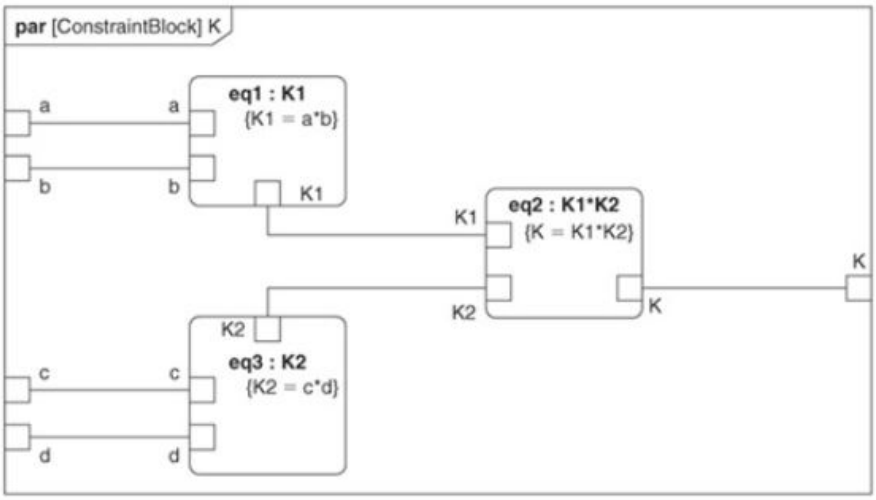


Abbildung 2.21: Darstellung eines Zusicherungsdiagrammes nach ([FMS08], Kap.7)

Aus Abbildung 2.21 ist die Verwendung des Konzeptes der Objektorientierung gut ersichtlich. Beispielsweise ist das Objekt *eq1* eine Instanz von dem vorher definierten Constraint *K1*. Dasselbe Prinzip wird bei den verbleibenden Elementen angewendet. Durch die Beziehungen, welche als Linien dargestellt sind, wird die gewünschte mathematische Gleichung nachgebildet. Das Zusicherungsdiagramm, welches in der Abbildung 2.21 abgebildet ist, ist in diesem Fall nur die Sicht auf das zugrundeliegende Systemmodell. Um den Nutzen von Systemmodellen zu erweitern, ist es möglich, von außen auf die Informationen im Systemmodell zuzugreifen. In diesem Fall wäre das der Zugriff auf die mathematische Gleichung, welche in Abbildung 2.21 grafisch dargestellt wird. Diese Gleichung kann deshalb beispielsweise über Schnittstellen an weiterführende Werkzeuge übertragen werden. Den Nutzen dieser Möglichkeit verdeutlicht Raimund Schlotmann, der Systems Engieering als die Basis eines Digitalen Zwillings bezeichnet. ([Sch18], S.38) Wird die Gleichung oder das Ergebnis davon in einem weiteren Entwicklungsschritt benötigt, kann über das Systemmodell darauf zugegriffen werden. Ein Zugriff auf diese Gleichung oder das Ergebnis, stellt zwar schon einen Nutzen dar, da die Gleichung nur einmal modelliert werden muss, allerdings entsteht ein viel größeres Nutzenpotential im weiteren Entwicklungsprozess. Im Falle der Gleichung würde eine Parameteränderung im Laufe des Entwicklungsprozesses eine Änderung des Ergebnisses hervorrufen. Diese Änderung kann mit Hilfe von Schnittstellen automatisch bei allen Verwendungen der Gleichung aktualisiert werden, was dazu führt, dass stets das Ergebnis der aktuellen Pa-

rameter verwendet wird. Diese Beschreibung ist ein kurzer Vorgriff auf den Abschnitt 2.4, um bereits an dieser Stelle Möglichkeiten des Konzeptes des Digitalen Zwillings in Verbindung mit dem modellbasierten Entwicklungsansatz zu zeigen.

Die nächste Gruppe von Diagrammen sind die Verhaltensdiagramme. Wie der Name schon verrät, wird mit diesen Diagrammtypen das Verhalten des Systems dargestellt.

Anwendungsfalldiagramm
Das Anwendungsfalldiagramm wird im Englischen als *Use Case Diagram* bezeichnet. In diesem Diagrammtyp wird dargestellt, welche Anwendungsfälle auftreten können. Ein Anwendungsfall entsteht dann, wenn das System von außen benutzt wird. Das bedeutet, bei Anwendungsfalldiagrammen sind immer Aktoren vorhanden, die das System benützen. Diese Akoren können reale Personen oder auch andere Systeme sein. Laut Alt ist das Anwendungsfalldiagramm einfach in der Darstellung und daher bei Projektbeteiligten beliebt. Allerdings besteht durch die Einfachheit des Diagrammtyps die Gefahr mehr in die Diagramme hineinzuinterpretieren als tatsächlich ausgesagt wird. ([Alt12], S.49)
In Abbildung 2.22 ist ein einfaches Anwendungsfalldiagramm grafisch veranschaulicht. Dargestellt werden in diesem Diagramm vier Anwendungsfälle eines Fahrzeuges durch Personen.

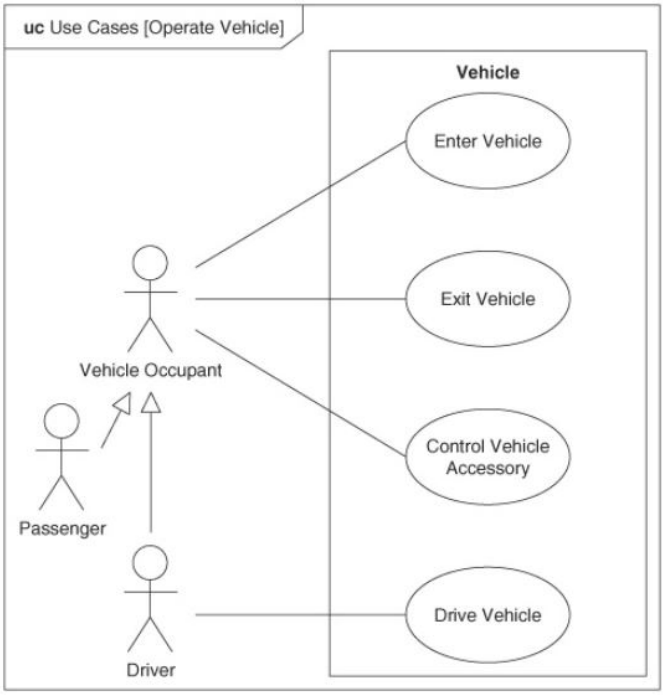


Abbildung 2.22: Darstellung eines Anwendungsfalldiagrammes nach ([FMS08], Kap.3)

Zudem sind die Aktoren abgebildet, die diese Anwendungsfälle hervorrufen. Durch die Darstellung der Anwendungsfälle wird das Bewusstsein geschaffen, welche Anwendungen

auftreten. Im Zuge der Erstellung des Diagrammes kann es leicht vorkommen, dass Anwendungsfälle auftreten, die vorher nicht berücksichtigt wurden.

Aktivitätsdiagramm
Ein weiteres Diagramm der Gruppe der Verhaltensdiagramme ist das Aktivitätsdiagramm. Wie der Name des Diagrammes schon verrät, dient diese Art von Diagramm dazu, die Aktivität eines Systems darzustellen. Eigentlich werden die Aktivitätsdiagramme mit Aktionen und Abfolgen dargestellt. Eine Aktivität ist in diesem Fall die Klasse einer Aktion. Typischerweise werden die Aktivitäten im Vorhinein definiert und die durch Instanzierung entstehenden Objekte als Aktionen zur Bildung eines Aktivitätsdiagrammes verwendet. In Abbildung 2.23 ist ein Beispiel eines Aktivitätsdiagrammes zu sehen. Die Kästchen mit den abgerundeten Ecken stellen die Aktionen dar. Durch die Pfeilrichtungen wird die Abfolge der Aktionen festgelegt.

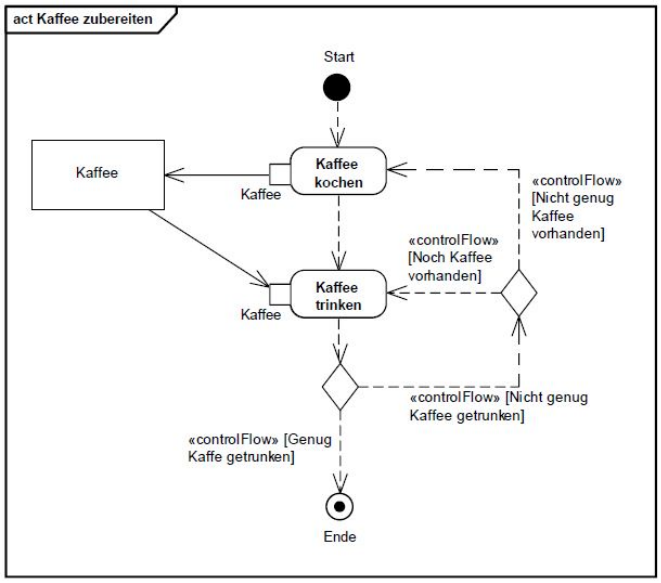


Abbildung 2.23: Darstellung eines Aktivitätsdiagrammes nach ([Alt12], S.55)

Zustandsdiagramm
Ein weiteres Diagramm der Gruppe der Verhaltensdiagramme ist das Zustandsdiagramm. In diesem werden die möglichen Zustände eines Systems dargestellt. Um von einem Zustand in den nächsten wechseln zu können, werden sogenannte Transitionen verwendet. Diese werden als Pfeile dargestellt. In Abbildung 2.24 ist ein Beispiel eines Zustandsdiagrammes dargestellt.
Die in Abbildung 2.24 dargestellten Kästchen mit den abgerundeten Ecken stellen die Zustände des Systems dar. Die Pfeile verdeutlichen die Transitionen. Deren Pfeilrichtung bestimmt von welchem Zustand, in welchen Zustand gewechselt werden kann.

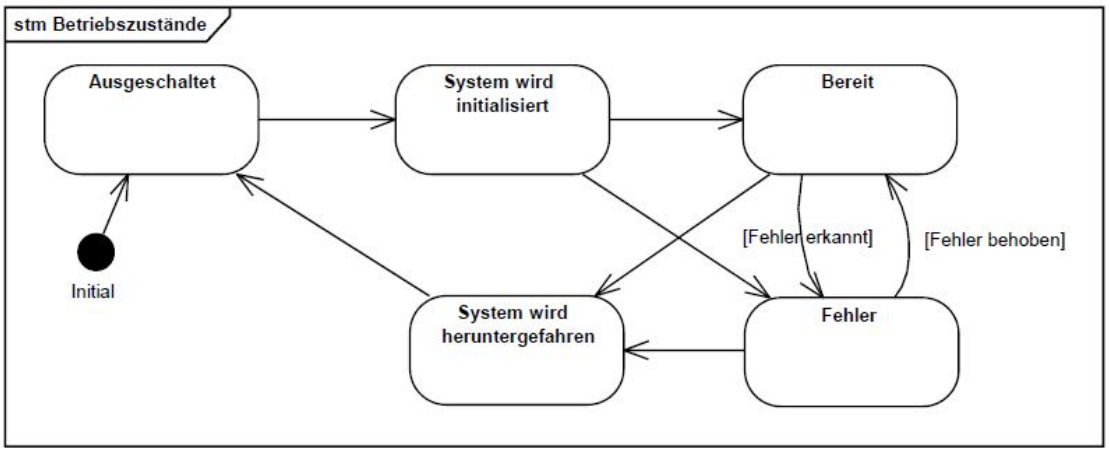


Abbildung 2.24: Darstellung eines Zustandsdiagrammes nach ([Alt12], S.59)

Sequenzdiagramm

Das letzte Verhaltensdiagramm ist das Sequenzdiagramm. Laut Alt dient das Sequenzdiagramm dazu, die Interaktionen und Kommunikationsabläufe zwischen den Systemkomponenten im zeitlichen Verlauf darzustellen. Die Systemkomponenten werden horizontal nebeneinander aufgereiht, die Zeitachse verläuft vertikal nach unten. Ein Sequenzdiagramm stellt immer nur ein bestimmtes Systemverhalten unter bestimmten Voraussetzungen dar. Es ist laut Alt daher praktisch unmöglich, alle erdenklichen Sequenzen in Diagrammen darzustellen. ([Alt12], S.53) Dennoch können solche Diagramme sinnvoll sein, um beispielsweise über bestimmte Sequenzen besser diskutieren zu können oder ein gemeinsames Verständnis dieser Sequenz zu fördern. Die wichtigsten Elemente von Sequenzdiagrammen sind die Blöcke, die die Elemente des Systems darstellen und die Funktionen, die den jeweiligen Systemelementen zugeordnet werden. In Abbildung 2.25 ist ein Sequenzdiagramm grafisch dargestellt. Die darin abgebildeten rechteckigen Kästchen stellen die Systemelemente dar. Die Beschriftungen über den Pfeilen verdeutlichen die Funktionen.

Anforderungsdiagramm

Das Anforderungsdiagramm gehört weder zu den strukturellen noch zu den Verhaltensdiagrammen. Es gehört zur Gruppe der sonstigen Diagramme und wurde für die SysML eingeführt. Das heißt in der UML gibt es keine Anforderungsdiagramme in dieser Form. Laut Alt wäre es grundsätzlich denkbar, die Informationen aus den Anforderungen auch mit Verhaltensdiagrammen darzustellen. Allerdings wird das in der Praxis bis jetzt kaum umgesetzt. Ein weiterer Grund für die textuelle Darstellung der Anforderungen ist, dass die textuelle und somit weniger formelle Form keine spezielle Vorbildung der Benutzer voraussetzt, um die Diagramme richtig zu interpretieren. Durch die textuelle Darstellung der Anforderungen geht die Möglichkeit der Maschinenlesbarkeit verloren, was einen Nachteil darstellt. ([Alt12], S.51)
Bei der Erstellung eines Anforderungsdiagrammes ist auf die im Kapitel 2.1 erläuter-

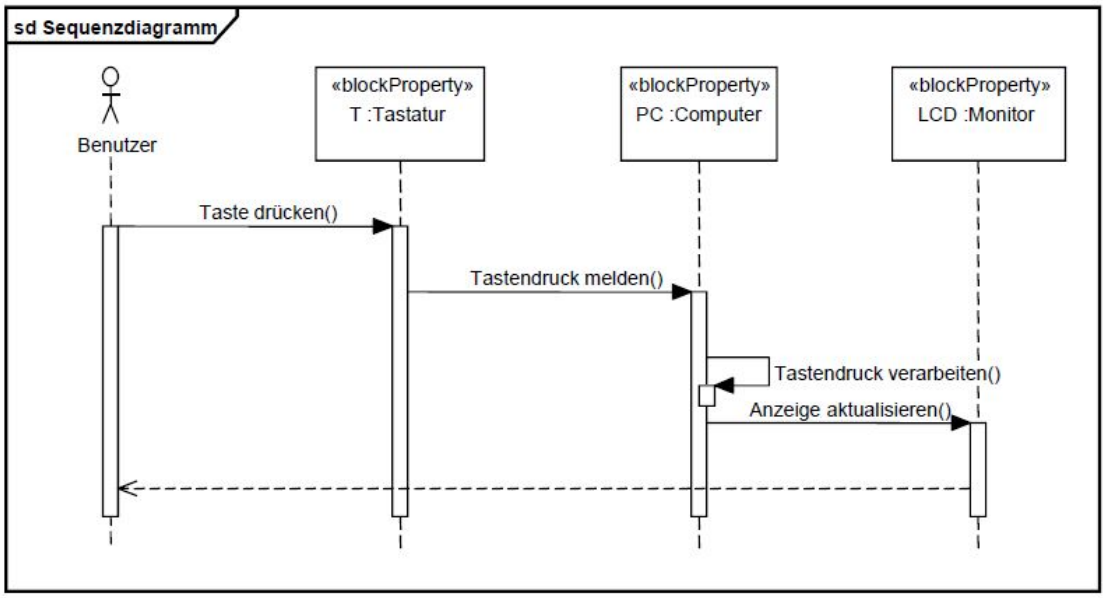


Abbildung 2.25: Darstellung eines Sequenzdiagrammes nach ([Alt12], S.53)

ten Vorgehensweisen zu achten. Neben der grundsätzlichen Vorgehensweise ist es außerdem unerlässlich, die Anforderungen eindeutig und widerspruchsfrei zu formulieren sowie die erwähnten Attribute zu definieren. Manche Werkzeuge bieten die Möglichkeit zum Importieren der Anforderungen über das RIF Format. In so einem Fall können die Anforderungen in einer speziell für das Anforderungsmangement entwickelten Software erstellt werden und im Anschluss importiert werden. Dieser Ansatz birgt allerdings die Gefahr, dass beispielsweise an unterschiedlichen Versionen gearbeitet wird oder, dass falsche Versionen importiert werden. Weiters ist auch der wirtschafliche Aspekt nicht zu vernachlässigen. Werden mehrere neue Softwaretools eingesetzt, um MBSE zu ermöglichen, steigen die Kosten natürlich dementsprechend. In der Abbildung 2.26 ist ein Beispiel eines Anforderungsdiagrammes grafisch dargestellt.
Mit dem Anforderungsdiagramm sind alle Diagramme der SysML erklärt. Neben all den vordefinierten Elementen ermöglicht die SysML durch sogenannte *Profile* und *Stereotypen* benutzerdefinierte Erweiterungen. Diese Erweiterungen werden bereits für die Erstellung der SysML aus der UML angewendet und können aber auch für branchenspezifische Belange angewendet werden. Das Ziel der Profile und Stereotypen ist es, die Sprache besser auf einen bestimmten Anwendungsfall oder eine bestimmte Anwendungsrichtung zuzuschneiden. In Abbildung 2.27 ist die grafische Darstellung der Erweiterung einer Klasse um einen Stereotyp zu sehen.
Durch die große Anzahl der Darstellungsmöglichkeiten in der SysML ist es gerade für die ersten Modelle schwierig, die richtigen Diagrammtypen und Elemente auszuwählen. Es erfordert viel Übung und Erfahrung, um ein Modell in einer Art und Weise aufzubauen, um es als Basis für einen Digitalen Zwilling zu verwenden. In der abschließenden Grafik zu Erklärung der SysML ist die Abstraktion von einem realen Objekt in die betreffenden

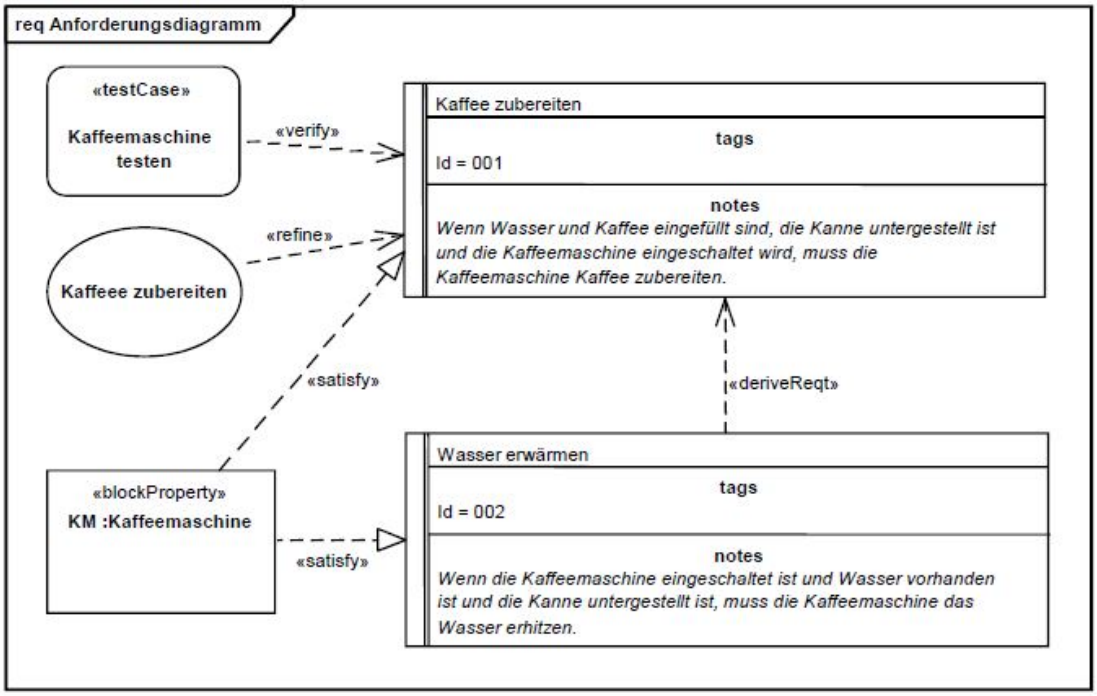


Abbildung 2.26: Darstellung eines Anforderungsdiagrammes nach ([Alt12], S.52)

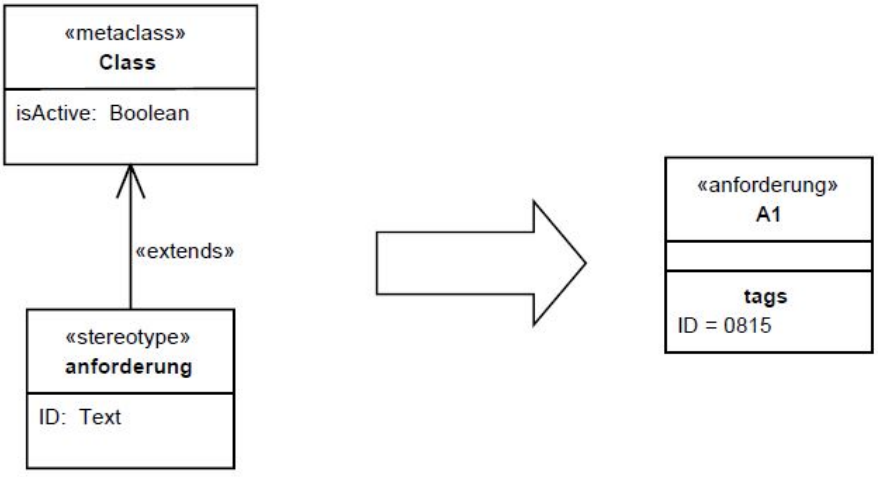


Abbildung 2.27: Darstellung des Profilmechanismus nach ([Alt12], S.27)

SysML-Elemente zu sehen. Diese Abbildung soll ein Beispiel darstellen, welche Elemente zur Modellierung verwendet werden können. In Abbildung 2.28 wird beispielsweise das Element Block verwendet, um den Flügel eines Flugzeugs darzustellen. Selbstverständlich ist diese Abbildung eine starke Vereinfachung eines Modells. Üblicherweise sind Darstellungen, die in der Praxis verwendet werden, sehr viel umfangreicher.

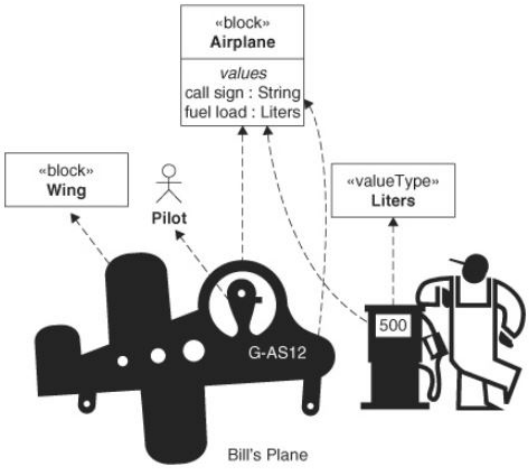


Abbildung 2.28: Beispiel einer Darstellung eines realen Objektes mit SysML-Elementen nach ([FMS08], Kap.4)

Die Abbildung 2.28 darf nicht als eigener Diagrammtyp verstanden werden. Diese Grafik gibt es in dieser Form nicht in der SysML. Sie dient lediglich dazu, zu veranschaulichen, welche Elemente der SysML zur Erstellung eines Modells verwendet werden können. Die verwendeten Elemente finden sich in den jeweiligen Diagrammtypen wieder.

CONSENS

CONSENS ist eine Sprache und Methode zugleich. Aus diesem Grund wird CONSENS in der Literatur oft als Spezifikationstechnik bezeichnet. In dieser Arbeit werden die Sprache und die Methode der Spezifikationstechnik mit CONSENS bezeichnet. Entwickelt wurde diese laut Kaiser auf Basis der Arbeiten von Kallmeyer, Frank und Gausemeier et al. [GEK01][Kal98][Gau+08] am Heinz Nixdorf Institut im „Sonderforschungsbereich 614“. ([Kai13], S.46) Mit CONSENS wird versucht, die Lücke zwischen den Anforderungen, die naturgemäß weit interpretierbar sind und den etablierten Spezifikationstechniken der einzelnen Disziplinen zu schließen. ([Zin13], S.67) Verglichen mit SysML, werden in der Sprache CONSENS nur sehr wenige Elemente verwendet, um ein Modell zu erstellen. In Abbildung 2.29 sind die Modellkonstrukte abgebildet, die zur Beschreibung einer Systemstruktur vorhanden sind.
Als Beispiel einer möglichen Verwendung der Modellkonstrukte ist in Abbildung 2.30 ein Umfeldmodell dargestellt, wobei die gelben Elemente für das Umfeld eines zu modellierenden Systems stehen.

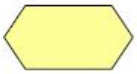
Modellkonstrukte	Graphische Notation	Semantik/Verfeinerung nach GEHRKE
Umfeldelement		Elemente außerhalb der Systemgrenze/ *keine*
System/ Systemelement		System entspricht dem zu entwickelnden Produkt und besteht aus Systemelementen/ Hardware-, Software- und logische Elemente
Energiefluss		Energiebeziehung zwischen zwei Elementen/ Energieart
Stofffluss		Stoffbeziehung zwischen zwei Elementen/ Stoffart
Informationsfluss		Informationsbeziehung zwischen zwei Elementen/ Informationsart
Logische Beziehung		Semantik ergibt sich durch die Bezeichnung der Beziehung

Abbildung 2.29: Modellkonstrukte zur Erstellung eines Systemmodells nach ([Kai13], S.47)

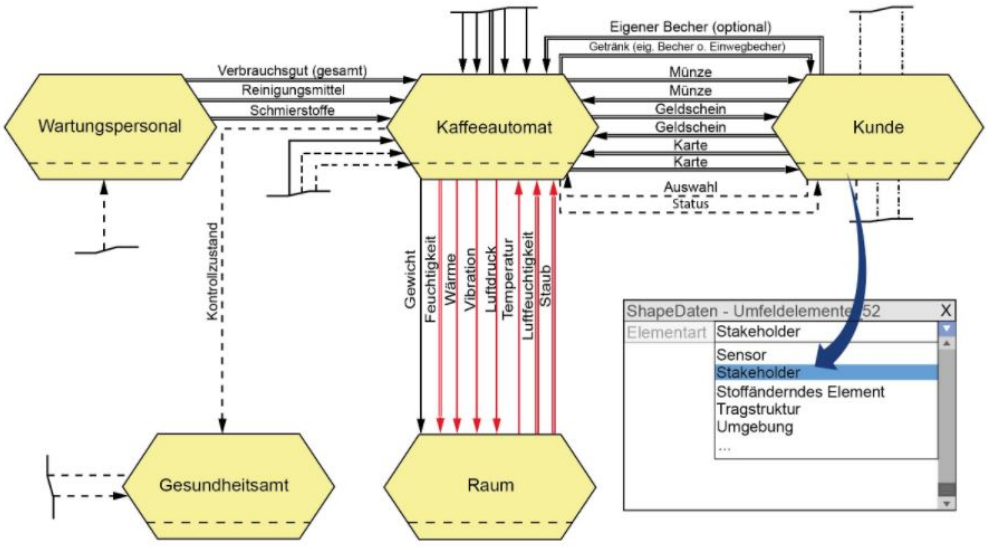


Abbildung 2.30: Beispiel eines Umfeldmodelles mit den Modellkonstrukten von CONSENS [Two]

Um welches System es sich handelt, ist aus dieser Darstellung nicht ersichtlich, aber in diesem Fall auch nicht relevant. Die Bedeutung der Beziehungen untereinander ist direkt aus der Beschreibung oder aus der Abbildung 2.29 zu entnehmen. Die Darstellung des Systems mit den Systemelementen erfolgt in der selben Art und Weise. Mit den Systemelementen werden die eigentlichen Komponenten (Mechanik, Elektronik, Software) des Systems modelliert. Die in Abbildung 2.29 dargestellten Elemente bilden das Grundgerüst der CONSENS Sprache und werden in bestimmten Werkzeugen um weitere Elemente erweitert. Diese Erweiterungen werden im Kapitel 4 näher erläutert. Die neben der Sprache CONSENS gleichnamige Methode wird im Abschnitt 2.3.5 näher erläutert.

2.3.5 Methoden zum Erstellen eines Systemmodells

Es gibt mehrere Methoden zum Erstellen eines Systemmodells. Da im Zuge dieser Masterarbeit Systemmodelle von mechatronischen Produkten erstellt werden, ist die Hauptanforderung der verwendeten Methode die Eignung zur Modellierung solcher. Die Methode soll dabei helfen im Modellierungsprozess den roten Faden zu behalten und zu verhindern, dass der Überblick verloren geht. Beispiele solcher Methoden sind in der Abbildung 2.9 zu finden. Im Zuge dieser Masterarbeit wird die Methode CONSENS verwendet. Diese besteht aus sechs Schritten und wurde explizit für die Entwicklung von mechatronischen Produkten entwickelt. Die Abbildung 2.31 stellt die sechs Schritte der Methode CONSENS dar.

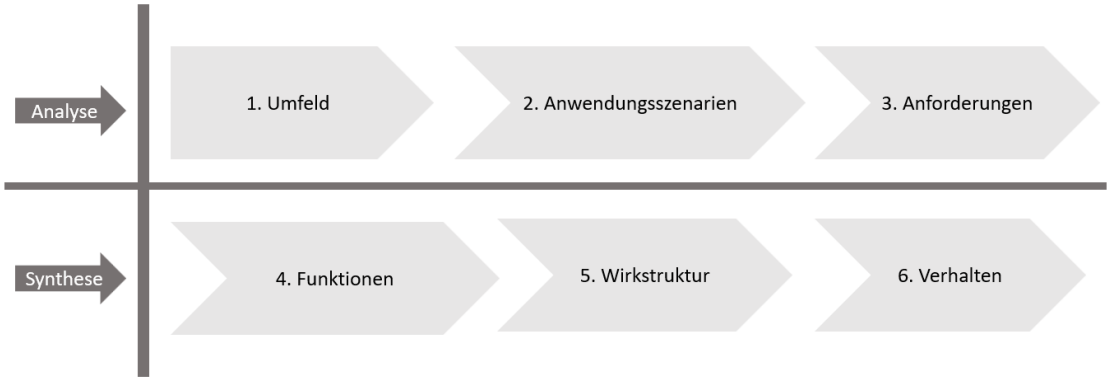


Abbildung 2.31: Sechs Schritte der Methode CONSENS in Anlehnung an ([GTS], S.38)

Die sechs Schritte der Methode CONSENS sind in zwei Bereiche zu unterteilen. Zuerst erfolgt die Analyse mit den dazugehörigen Schritten Umfeld, Anwendungsszenarien und Anforderungen. Der zweite Bereich ist die Synthese, diese setzt sich zusammen aus den Funktionen, der Wirkstruktur und dem Verhalten. Die Methode wird im Zuge des Modellierungsprozesses mehrmals durchlaufen, da im Laufe eines Entwicklungsprozesses neue Erkenntnisse und Informationen auftauchen. Im Folgenden wird die Bedeutung der einzelnen Schritte erklärt, um ein Grundverständnis für die Methode zu erlangen. Bei der Beschreibung der einzelnen Schritte ist zu beachten, dass diese nur im Zusammenhang mit einem Werkzeug und einer Sprache ausgeführt werden. Das bedeutet beispielsweise

nach abgeschlossener Umfeldanalyse liegen die aus dieser Analsyse stammenden Informationen bereits im Systemmodell vor. Im Zuge des Durchlaufens der Methode wird daher gleichzeitig das Systemmodell erstellt. In Kapitel 4 wird die praktische Anwendung dieser Methode dargestellt.

1. Umfeld
Im ersten Schritt der CONSENS Methode wird das Umfeld des zu entwickelnden Systems betrachtet. Im Zuge dieser Betrachtungen werden Umfeldeinflüsse aufgedeckt die bei isolierter Betrachtung des Systems nicht aufgetaucht wären. Beispiele für Einflüsse aus dem Umfeld des Systems sind die Energieversorgung, der Abtransport von Gütern, die gesetzlichen Bestimmungen in der Kennzeichnung und die Zulassung des Systems.

2. Anwendungsszenarien
Nach der Umfeldanalyse werden die möglichen Anwendungsszenarien des Systems erarbeitet. Durch diesen Schritt wird darauf aufmerksam gemacht, welche möglichen Szenarien für ein System relevant sein können. Durch das bewusste Suchen nach Anwendungsszenarien wird vermieden, dass bestimmte Szenarien im Entwicklungsprozess wenig oder keine Beachtung finden. Des Weiteren führen verschiedene Anwendungsszenarien zu verschiedenen Anforderungen.

3. Anforderungen
Im Gegensatz zu den vorhergehenden Schritten haben die Anforderungen mitunter direkte Auswirkungen auf die Funktionen eines Systems. Die im Abschnitt Anforderungsmanagement beschriebene Vorgehensweise zum Finden und Dokumentieren von Anforderungen ist in diesem Schritt anzuwenden. Die Anforderungen bilden die Grundlage für die Entwicklung des Systems. Die Formulierung der Anforderungen und der notwendige Inhalt dieser ist ebenfalls dem Abschnitt Anforderungsmanagement zu entnehmen. Folgender Satz ist ein Beispiel für eine Anforderung.
Anforderung: *Das Bauteil muss in 10 s von A nach B transportiert werden.*
In diesem Fall handelt es sich um die Beschreibung der Anforderung. Weitere notwendige Attribute sind in diesem Fall nicht aufgeführt. Im weiteren Schritt der CONSENS Methode werden aus den Anforderungen die Funktionen abgeleitet.

4. Funktionen
Im vierten Schritt der Methode werden die Funktionen des Systems untersucht. Jedes technische System kann in Funktionen beschrieben werden. Funktionen beschreiben lediglich die Funktion, die erfüllt werden soll, nicht aber die technische Umsetzung. Durch die Trennung von Funktion und Umsetzung kann ohne detaillierte technische Kenntnisse die Funktionsweise eines Systems abgebildet werden. Sind die zur Realisierung eines Systems notwendigen Funktionen festgelegt, können die jeweiligen Fachrichtungen mit der Umsetzung dieser Funktionen fortfahren. Abgeleitet werden die Funktionen zu großen Teilen aus den vorher definierten Anforderungen. Der Funktionsumfang kann, wie auch der Anforderungsumfang, während des Entwicklungsprozesses wachsen, da die CONSENS Methode ein iterativer Prozess ist. Am Beispiel der obigen Anforderung ist eine dazuge-

hörige Funktion folgende:
Funktion: *Bauteil transportieren* mit der Unterfunktion: *Motor ansteuern*
Auf Basis der Funktionen können die Fachrichtungen mit der jeweiligen Umsetzung der Funktion beginnen. Klarerweise erfordert die finale Umsetzung der Funktion mehr Informationen als bisher angeführt, allerdings lassen diese Funktionen erste Schritte in der Entwicklung der E-Technik oder der Software zu, ohne ein fertiges CAD Modell vorliegen zu haben. Im Anschluss an die Funktionsbeschreibung folgt die Wirkstruktur.

5. Wirkstruktur
Die Wirksturktur stellt erstmalig im Verlauf der CONSENS Methode die Umsetzung der Funktionen in den Mittelpunkt. In diesem Schritt werden die logischen und physischen Elemente festgelegt, die zur Umsetzung der Funktionen dienen. Der Unterschied zwischen den logischen und physischen Elementen ist, dass erstere eine allgemeine Bezeichnung haben und daher keinem realen Objekt direkt zugeordnet werden können. Eine mögliche Bezeichnung für ein logisches Element ist *Motor*. Mit dieser Bezeichnung ist zwar im groben Sinn definiert was gemeint ist, aber es ist keine genaue Information vorhanden. Im Gegensatz dazu liefert das physische Element die genaue Bezeichnung welches Objekt gemeint ist. In diesem Fall lautet die Bezeichnung anstatt *Motor* beispielsweise *Motor E713 BT0123846*. Damit ist das Objekt eindeutig identifiziert und es können weitere Informationen dazu eingeholt werden. Die Wirkstruktur wird gleichermaßen für alle Fachdisziplinen (Mechanik, E-Technik, Software) aufgebaut. Eine vollständige Wirkstruktur stellt alle Elemente sowie die Verbindungen eines Systems dar. Die durch die Wirkstruktur entstehende Darstellung kann auch als mechatronische Beschreibung des Systems bezeichnet werden.
Der letzte Schritt in der CONSENS Methode ist das Verhalten.

6. Verhalten
Um das Verhalten eines Systems zu modellieren, werden die vorher definierten Funktionen verwendet. Die Darstellung des Verhaltens kann in Form von Funktions- oder Zustandsdiagrammen erfolgen. Funktionsdiagramme dienen als wichtige Basis für die Softwareabteilungen.

Die Methode CONSENS ist als iterativer Prozess zu betrachten, der mit jedem Iterationschritt Information hinzugewinnt. Die Sprache, mit der die Methode CONSENS umgesetzt werden kann, ist einerseits die Sprache CONSENS selbst, aber auch SysML bietet, durch das Profil SysML4CONSENS, die Möglichkeit ein Systemmodell nach der Methode CONSENS zu erstellen.

2.3.6 Werkzeuge zum Erstellen eines Systemmodells

Nach der näheren Erläuterung von Sprachen und Methoden fehlt noch die letzte der drei Komponenten zur Erstellung eines Systemmodells, das Werkzeug. Mit dem Begriff

Werkzeug ist in diesem Fall ein Softwaretool gemeint, das die Möglichkeit bietet, ein Systemmodell in der gewünschten Sprache mit der gewünschten Methode zu erstellen. Gemeinsamkeiten weisen die Softwaretools bei der Verwendung von gängigen Modellierungssprachen auf. Nennenswerte Unterschiede sind beispielsweise in der Umsetzung des User Interface, den vorhandenen Funktionen oder den vorhandenen Schnittstellen zu finden. Das bedeutet, dass das Anwendungsgebiet direkten Einfluss auf die Auswahl des Werkzeuges hat.

Durch den modellbasierten Ansatz ist es beispielsweise möglich, schneller ein gemeinsames Systemverständnis zu erlangen oder die im Systemmodell hinterlegten Daten weiterzuverwenden. Durch die Integration des modellbasierten Ansatzes in den Produktentwicklungsprozess, genauer gesagt in den virtuellen Produktentwicklungsprozess, können diese Potentiale entfaltet werden. Auf die virtuelle Produktentwicklung wird im nächsten Kapitel (2.4) eingegangen.

2.4 Modellbasierte virtuelle Produktentwicklung

Die modellbasierte virtuelle Produktentwicklung stellt eine Möglichkeit dar, erstellte Modelle im gesamten Produktentwicklungsprozess zu verwenden. Eigner definiert den Begriff *Modellbasierte Virtuelle Produktentwicklung* wie folgt: „Modellbasierte Virtuelle Produktentwicklung (MVPE) ist die durchgehende, rechnerunterstützte, formale Modellbildung und Dokumentation entlang aller entwicklungsrelevanten Phasen des Produktlebenszyklus mit der Zielsetzung der Weitergabe des Modells in die nächste Entwicklungsphase sowie der Weiterverwendung dieser Modelle für Simulation, Validierung und Verifikation. Ziel ist die frühe Erarbeitung des Produkt- und Produktionswissens und damit das frühzeitige Optimieren von Produkteigenschaften im Sinne einer ganzheitlichen Optimierung des gesamten Produktlebenszyklus sowie die drastische Reduzierung von physischen Prototypen “ ([Eig14], S.9).

Neben der Darstellung des theoretischen Hintergrundes der modellbasierten virtuellen Produktentwicklung, soll dieser Abschnitt die Zusammenhänge des Anforderungsmanagements und des Model Based Systems Engineering mit der modellbasierten virtuellen Produktentwicklung erläutern.

2.4.1 Produktentwicklungsprozess in der modellbasierten virtuellen Produkentwicklung

Laut Eigner unterstützt die Virtuelle Produktentwicklung (VPE) alle Phasen der Produktentwicklung. Der Produktentwicklungsprozess umfasst die Phasen von der Angebotsbearbeitung bis zu den Fertigungs- und Montageprozessen. Weiters wird auch das Management der Informationen, die im Laufe der Produktentwicklung anfallen, als Teil der VPE verstanden. Prozesse und IT-Lösungen zielen darauf ab, die Entwicklungsstufen

zu beschreiben, zu dokumentieren und für die nächste Entwicklungsphase zur Verfügung zu stellen. Im Zuge des Managements von Informationen und den dazugehörigen IT-Lösungen wird von einer IT-Prozesskette gesprochen. Eine solche Prozesskette wird als modellbasiert bezeichnet, wenn die Beschreibung der jeweiligen Produktentwicklungsphase in einer formalen Sprache erfolgt. ([Eig14], S.7)
In Abbildung 2.32 ist der gesamte Produktlebenszyklus dargestellt. Die Produktentwicklung stellt den Beginn des Produktlebenszyklus dar. Im Zuge der modellbasierten virtuellen Produktentwicklung werden IT-Lösungen und formale Sprachen verwendet, um die anfallenden Informationen in standardisierten Datenformaten zu speichern. Die verwendeten Datenformate müssen die Weiterverwendung der Daten in den darauffolgenden Entwicklungphasen ermöglichen. Im Zuge dieser Arbeit wird auf die ersten drei Phasen des Produktlebenszyklus eingegangen. Auf die erste Phase, der Anforderungsdefinition, folgt die Produktplanung und die Konzeptphase. Diese drei Phasen sind einerseites Teil des Produktlebenszyklus aber auch Teil des integrierten Produktentwicklungsprozesses. In weiterer Folge wird daher nur mehr von den Phasen des Produktentwicklungsprozess gesprochen. Die erste Phase der Produktentwicklung stellt die Anforderungsdefinition dar. In dieser Phase wird das in Abschnitt 2.1 beschriebene Anforderungsmanagement angewendet. Das Prinzip der Weiterverwendung der Daten in den darauffolgenden Phasen ist eine zentrale Anforderung der modellbasierten virtuellen Produktentwicklung und muss über den gesamten Produktlebenszyklus gewährleistet sein.
Die Konzepte des Anforderungsmanagements finden sich hauptsächlich in der ersten Phase des Produktentwicklungsprozesses wieder. Eine strike Einschränkung auf die Phase der Anforderungsdefinition wäre allerdings falsch, da Tätigkeiten wie das Pflegen und Verwalten von Anforderungen oder die Beurteilung von Risiken nicht nur in der Phase der Anforderungsdefinition stattfinden.
Um die in der Definition von Eigner erwähnte *formale Modellbildung* bewerkstelligen zu können, wird auf das Konzept des Model Based Systems Engineering zurückgegriffen. Das Systemmodell als Herzstück des MBSE kann als formales Modell verstanden werden, da es die Weiterverwendnung von Daten in nachfolgenden Entwicklungsphasen ermöglicht.

2.4.2 Modelle in der virtuellen Produktentwicklung

Die Verwendung von Vorgehensweisen und Modellen in der Produktentwicklung ist nicht neu. Jede Fachdisziplin verwendet jene, die ihren Anforderungen am besten entsprechen. Zur Umsetzung der modellbasierten virtuellen Produktentwicklung ist es erforderlich, disziplinübergreifende Vorgehensweisen und Modelle zu verwenden. Der Begriff *disziplinübergreifendes Modell* beschreibt die gemeinsame Betrachtung von mechanischen, elektronischen, softwaretechnischen und weiteren im Produktentwicklungsprozess auftretenden Aspekten in einem Modell. Das disziplinübergreifende Vorgehen verbindet die verschiedenen Aspekte der Produktentwicklung und führt diese in ein Modell zusammen. Dieses Modell dient als Speicherort und Quelle von Informationen für alle Beteiligten. Ein Systemmodell, welches im Zuge von MBSE erstellt wird, ist beispielsweise ein disziplinübergreifendes Modell, welches zusätzlich die Möglichkeit bietet, die Informationen an nachgelagerte Entwicklungsphasen weiterzugeben.

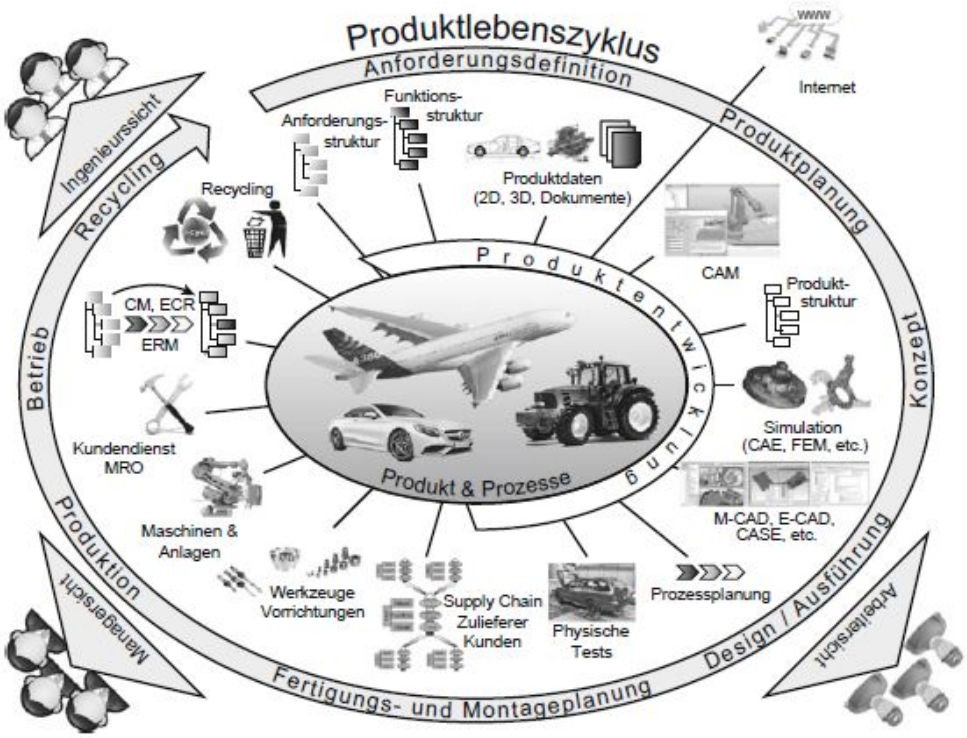


Abbildung 2.32: Integrierte Produktentwicklung im Produktlebenszyklus nach ([Eig14], S.8)

Als Vorgehensmodell der virtuellen Produktentwicklung kann das in Abbildung 2.33 dargestellte MVPE-Vorgehensmodell von Eigner, Gilz und Zafirov genannt werden.
Im Zuge dieser Arbeit wird die interdisziplinäre Systementwicklung näher betrachtet, welche die linke Seite des MVPE-Vorgehensmodells darstellt. Die interdisziplinäre Systementwicklung kann in vier Teilbereiche gegliedert werden. Beginnend mit den Anforderungen, folgt die Funktion des Systems. Bis zu diesem Teil des Vorgehensmodells wird die technische Umsetzung nicht beachtet. Erst mit den Elementen der logischen Architektur werden die notwendigen Elemente zur Umsetzung der vorher definierten Anforderungen und Funktionen festgelegt. Die Elemente der logischen Architektur weisen keine genaue Beschreibung auf, die es ermöglichen würde, dieses Element eindeutig zu identifizieren. Beispiele für die Bezeichnung dieser Elemente sind “Motor“ oder “Förderband“. Die Bezeichnung erlaubt ein ungefähres Bild der technischen Umsetzung, allerdings fehlen die Details. In der Konzeptphase erlaubt diese Betrachtung schnelle Iterationen ohne Zeitverlust durch Erstellen von Detaillösungen, die im Nachhinein verworfen werden. Im letzten Schritt der interdisziplinären Systementwicklung werden die physikalischen Elemente beschrieben. Diese weisen die bei den Elementen der logischen Architektur fehlenden Details auf.
Die beschriebene Vorgehensweise deckt sich mit Teilen der Methode CONSENS aus Ab-

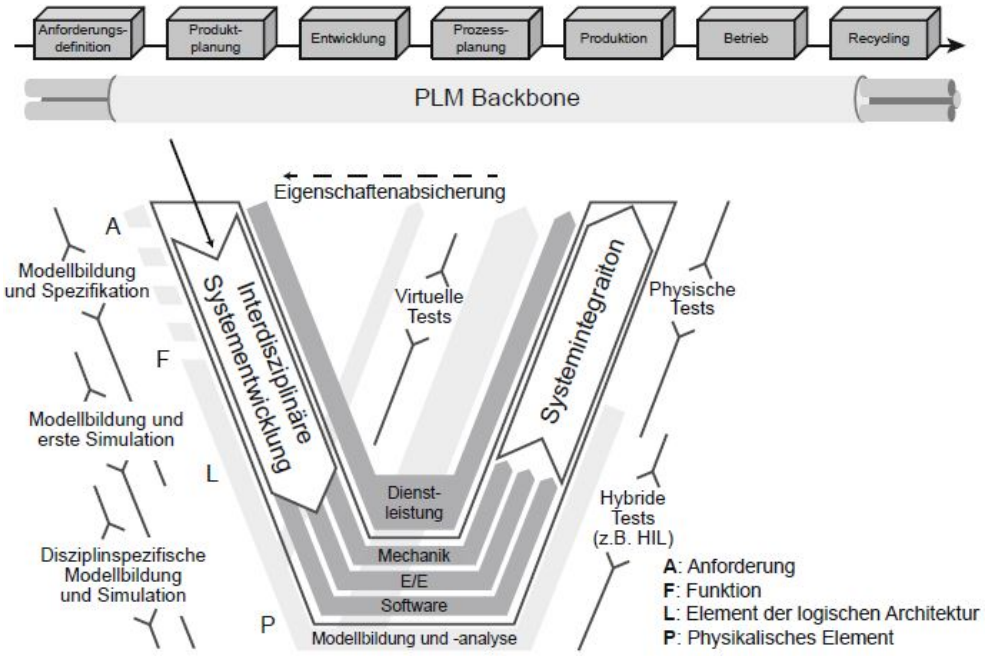


Abbildung 2.33: MVPE-Vorgehensmodell nach ([EGZ12], S.1670)

schnitt 2.3.5.

2.4.3 Verwendung der Modelle

Im Zuge der modellbasierten virtuellen Produktentwicklung können Modelle in unterschiedlicher Art und Weise verwendet werden. Wird die Betrachtung auf die Systemmodelle beschränkt, die im Zuge des MBSE Ansatzes entstehen, ergeben sich folgende Verwendungsmöglichkeiten:

- Speichern von Informationen, die im Entwicklungsprozess anfallen
- Gemeinsame Basis für Kommunikation über alle Fachdisziplinen hinweg
- Single Source of Truth
- Quelle für Informationen für alle Projektbeteiligten
- Erlangen eines gemeinsamen Systemverständnisses
- Weiterverwendung der Informationen in den nächsten Entwicklungsphasen
- Basis für einen digitalen Zwilling
- Schnittstelle für den Zugriff auf Informationen, die in Datenbanken gespeichert sind (z.B: CAD-Dateien)

- Bereitstellung von Parametern des Systems
- Bereitstellung von Informationen für die Simulation
- Verwendung ähnlicher Module für andere Systeme

Ein Systemmodell ist demzufolge mehr als nur eine grafische Darstellung von Informationen. Bei richtiger Anwendung stellt ein Systemmodell den Kern eines Projektes dar. Welche Anforderungen ein Systemmodell erfüllen muss, hängt von der Tätigkeit des Unternehmens und des Anwendungsgebietes ab. In dieser Arbeit wird ein Systemmodell erstellt, welches die Anforderungen eines Sondermaschinenbauers erfüllen muss.
Für die erfolgreiche Umsetzung der modellbasierten virtuellen Produktentwicklung ist daher darauf zu achten, dass die drei Hauptbestandteile, welche Einfluss auf ein Systemmodell haben (Abbildung 2.12), im Kontext der jeweiligen Anwendung untersucht und ausgewählt werden.

Eine Grundvoraussetzung zur Durchführung einer modellbasierten virtuellen Produktentwicklung ist ein, den Anforderungen entsprechendes Systemmodell und eine klare Anforderungsdefinition. Am Anfang der Produktentwicklung ist das Systemmodell ohne Informationen. Beginnend mit den Anforderungen werden dem Systemmodell im Laufe der Entwicklung Informationen hinzugefügt, welche im weiteren Entwicklungsprozess weiterverwendet werden können.

3 Aktueller Forschungsstand

In diesem Kapitel wird der aktuelle Forschungsstand im Bereich des MBSE und der darauf aufbauenden modellbasierten virtuellen Produktentwicklung näher erläutert. Dazu werden Beispiele von bereits existierenden Projekten verwendet. Darüber hinaus wird näher darauf eingegangen, wie die Informationen aus einem Systemmodell in Simulationswerkzeugen mit Hilfe von Schnittstellen verwendet werden können. Abschließend wird dargestellt, wie das Systemmodell als Basis eines Digitialen Zwillings fungiert.

3.1 Praktische Verwendung des MBSE Ansatzes

Erste praktische Verwendungen eines Systems Engineering Ansatzes fanden in der Luft- und Raumfahrtindustrie statt. Diese Branchen müssen das Zusammenspiel der einzelnen Fachdisziplinen perfekt beherrschen, da kleinste Fehler in Katastrophen enden können. Aus diesem Grund sind die Luft- und Raumfahrtindustrie Vorreiter auf diesem Gebiet. Durch die fortschreitende Verschmelzung der Fachdisziplinen steigt der Bedarf an einer Lösung für eine bessere Zusammenarbeit auch in Brachen abseits der Luft- und Raumfahrtindustrie.
Als bekanntes Forschungsprojekt in diesem Bereich ist das mecPro^2 Projekt zu nennen. In diesem wurden anhand von zwei Demonstratoren Erkenntnisse über den modellbasierten Enwicklungsprozess gewonnen. Mit Demonstrator 1 wurde die Entwicklung cybertronischer Produkte in den Mittelpunkt gestellt. Als Anwendungsbeispiel wurde das System „Autonomes Parken“ verwendet. Das Ziel dieses Anwendungsbeispieles war die praktische Umsetzung der Integration des Modells in ein Product Lifecycle Management (PLM)-System aufzuzeigen. Die Erkenntnisse aus Demonstrator 1 bestätigen den Nutzen der Modellintegration in das PLM-System. Durch die Integration wird das Zusammenwachsen der Disziplinen schon in den frühen Phasen gefördert.
Demonstrator 2 verwendete als Anwendungsbeispiel eine „Zylinderkopfproduktion“. Mit diesem Demonstrator wurde die kollaborative Entwicklung von Produkten und Produktionssystemen erforscht. Die Erkenntnisse aus Demonstrator 2 bestätigt die Möglichkeit des Einsatzes von MBSE in der frühen Phase der Produktentwicklung. Weiters wurde in diesem Demonstrator auch die Möglichkeit aufgezeigt, eine automatisierte Simulationsmodellerzeugung durchzuführen. ([MH17], S.227)

3.2 Verknüpfung des Systemmodells mit Simulationswerkzeugen im Produktentwicklungsprozess

Ein wichtiger Aspekt eines Systemmodells ist die Möglichkeit der Verknüpfung mit anderen Softwarewerkzeugen und Modellen. Diese Verknüpfung erfolgt in der Regel mit standardisierten Schnittstellen. Laut Müller und Kirsch wird in Fachkreisen die Integration möglichst vieler PLM-Informationen als sinnvoll erachtet. Zu diesen Informationen zählen beispielsweise Testberichte, Lasten - und Pflichtenhefte wie auch dokumentenbasierte Spezifikationen. Auch für Modelle aus den nachgelagerten Phasen im Entwicklungsprozess wie CAD-Modelle oder Stücklisten soll die Möglichkeit der Integration bestehen. Laut Müller und Kirsch treten bei der Verknüpfung der Daten aus dem Systemmodell mit anderen Werkzeugen folgende Probleme auf:

- Auf Ebene der Werkzeuge zur Erstellung eines Systemmodells, sind kaum prozessorientierte Funktionen vorhanden. Aus diesem Grund sehen die Autoren die Verknüpfung der Modelle im PLM-System, welches über diese Funktionen verfügt, als sinnvoll.
- Weiters liegen Dokumente und Modelle aus dem Produktlebenszyklus nicht zwangsläufig in der jeweiligen Modellierungssprache vor.
- Als letzten Punkt führen Müller und Kirsch an, dass die Systemmodelle mit Entwicklungsinformationen versehen werden sollen, die das Kompetenzspektrum des Systems Engineering möglicherweise nicht abdeckt. ([MK17], S.177)

Im mecPro^2 Foschungsprojekt wurde eine mögliche Verknüpfung von Systemmodellen mit anderen Werkzeugen wie in Abbildung 3.1 dargestellt. In diesem Fall handelte es sich um die Integration des SysML Modells in ein PLM-System. Neben der Verknüpfung mit einem PLM-System können die Daten des Systemmodells mit Simulationswerkzeugen verbunden werden. Gängige Simulationswerkzeuge sind beispielsweise Matlab oder Modelica. Laut Hanukaev u. a. muss bei der Verbindung von Systemmodell und Simulationsmodell darauf geachtet werden, dass der Aufwand der Modellierung auf Grund der simulationspezifischen Details nicht unverhältnismäßig ansteigt. Als Beispiel wird hierbei die Modelica-Transformation *SysML4Modelica* genannt. Das Übertragen von anwendungsfallspezifischen Informationen aus dem Systemmodell in eine Simulationsumgebung, um damit das Simulationsmodell zu parametrisieren, wird als weiterer Ansatz beschrieben. Im Rahmen des Foschungsprojektes mecPro^2 wurde ein weiterer Ansatz, das Sychronisationskonzept, angewendet. ([Han+17], S.120)
In Abbildung 3.2 sind die Hauptbestandteile des Synchronisationskonzeptes dargestellt. Das Systemmodell und das Simulationsmodell weisen inhaltliche Abhängigkeiten auf, können aber auf Grund der unterschiedlichen Modellierungssprachen nicht direkt miteinander verknüpft werden. Die in der Zwischenschicht dargestellten Modelle *SYS‘ und SIM‘* stellen die für die Simulation relevanten Bereiche der Modelle dar. Die direkte Verbindung von *SYS‘ und SIM‘* erfolgt im Verknüpfungsglied. Für das Forschungsprojekt mecPro^2 wurden sowohl für das Systemmodell als auch für die Zwischenschicht SysML

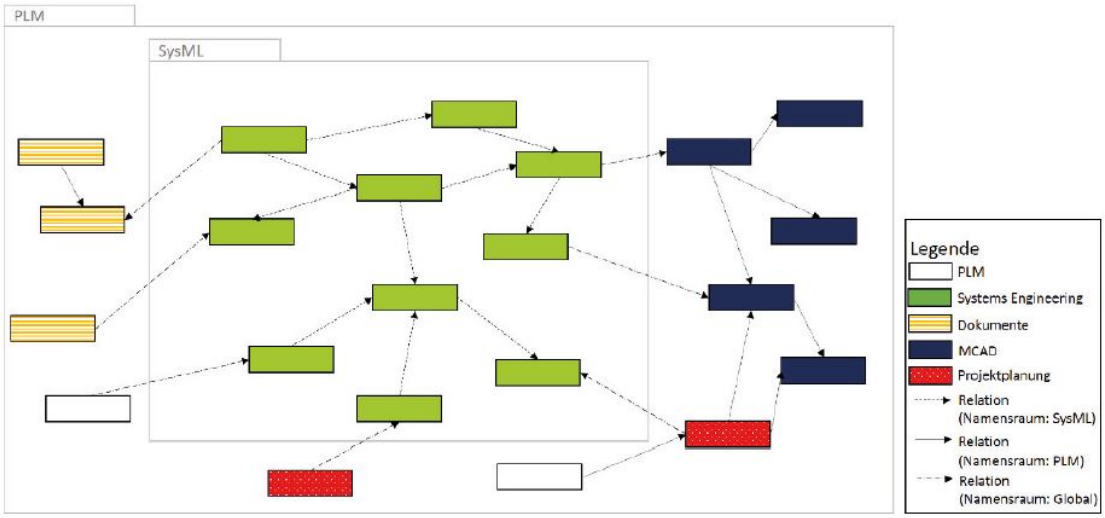


Abbildung 3.1: Verknüpfung von PLM-Objekten im Systemmodell nach ([MK17], S.180)

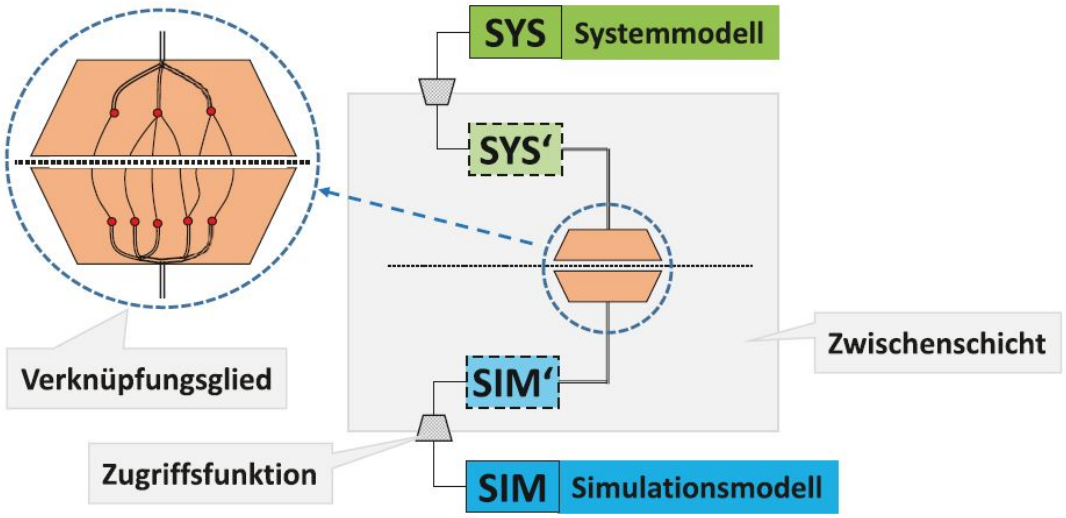


Abbildung 3.2: Bestandteile des Synchronisationskonzepts nach ([Han+17], S.121)

als Modellierungssprache verwendet. Zur Demonstration der Funktionsweise wurde eine cybertronische Parkhausschranke modelliert. Das Synchronisationskonzept ist durch die Einführung der Zwischenschicht kompatibel zu weiteren Sprachen. ([Han+17], S.121 ff.)

3.3 Systemmodell als Basis für einen Digitalen Zwilling im Entwicklungsprozess

Durch das disziplinübergreifende Systemmodell, das sämtliche für die Entwicklung relevante Informationen direkt enthält, oder beispielsweise über Links zu diesen Informationen führt, ergeben sich Möglichkeiten, die bei einem dokumentenbasierten Ansatz nicht möglich sind. Hierzu zählt beispielsweise die automatisierte Verwendung von Informationen im weiteren Entwicklungsprozess oder das Filtern nach bestimmten Informationen aus dem Systemmodell.
Laut Schlotmann stellt die Basis für Digitale Zwillinge das Systems Engineering dar. Der Autor spricht in diesem Zusammenhang nicht von einem vollständig digitalen Abbild, sondern von einer „Lebenslaufakte “, die die Struktur aller Elemente des Produktes beinhaltet und die gesamten relevanten Informationen in Form von Dokumenten mit den Elementen in Verbindung bringt. Diese Form des digitalen Zwillings wird vom Autor als „Digitaler Informationszwilling“ bezeichnet. ([Sch18], S.38)
Das Ziel eines Digitalen Zwillings ist laut Boschert und Rosen das Bereitstellen von Lösungen für unterschiedliche Ziele und Fragen. Diese können in allen Phasen des Entwicklungsprozesses auftauchen, wobei der größte Nutzen in den späteren Phasen zu erwarten ist. Um das zu erreichen, muss ein Digitaler Zwilling und dessen Funktionen, von Beginn des Entwicklungsprozesses an geplant sein. Laut den Autoren benötigt jeder Digitale Zwilling eine Struktur, die zur Erreichung bestimmter Ziele und zur Beantwortung von Fragen verhilft. Diese Struktur legt somit fest, für welche Zwecke der Digitale Zwilling verwendet werden kann. Ein weiterer wichtiger Aspekt ist der Informationsaustausch. Hierbei kann mit Hilfe durchgehender Anwendung von MBSE die Struktur eines Digitalen Zwillings mit den notwendigen Modellen und Daten befüllt werden. ([BR16], S.67 ff.)
Damit ein Digitaler Zwilling funktionieren kann, braucht dieser die notwendigen Informationen. Diese können teilweise aus schon vorhandenen IT-Systemen (e.g. PLM) oder aus Systemmodellen bezogen werden.

4 Vorgehensweise und Umsetzung

In diesem Kapitel wird zu Beginn die Vorgehensweise zur Beantwortung beider Forschungsfragen beschrieben. Danach wird die Auswahl der Sprache, der Methode und des Werkzeuges näher erläutert. In weiterer Folge wird die Erstellung zweier Systemmodelle mit den ausgewählten Komponenten dargestellt. Abschließend wird die Umsetzung der Verknüpfung von Systemmodell und Simulation im Zuge der Anforderungsüberprüfung veranschaulicht.

4.1 Vorgehensweise

Zur Beantwortung der Forschungsfragen wurde die Methode der angewandten Forschung gewählt. Die Gründe für diese Wahl sind, dass die Aufgabenstellung einerseits von einem Unternehmen stammt und andererseits die Forschungsfragen durch Darstellung von möglichen Lösungswegen beantwortet werden können. Auf Basis dieser Forschungsmethode wird die Beantwortung der Forschungsfragen mit praxisnahen Beispielen durchgeführt. Dazu wird zu Beginn die Auswahl der notwendigen Faktoren (siehe auch Abschnitt 2.3.3) zur Erstellung eines Systemmodells getroffen. Die Anforderungen zur Auswahl beruhen einerseits auf der Technik der Produkte und andererseits auf dem Einsatzgebiet. Als Quellen der Anforderungen dienen zum einen Gespräche mit Mitarbeitern aus den Bereichen Technik und Projektmanagement, zum anderen werden Anforderungen auf Basis eigener Betrachtungen der Produkte und des Einsatzgebietes ermittelt. Im weiteren Verlauf der Arbeit werden zwei Systemmodelle erstellt, mit denen verdeutlicht werden soll, dass die Auswahl der Faktoren für die Firma Fill geeignet sind. Durch den Prozess des Erstellens der Systemmodelle kann überprüft werden, ob die festgelegten Anforderungen an die Faktoren erfüllt werden.
Zur Beantwortung der zweiten Forschungsfrage wird in Anlehnung an das Synchronisationskonzept (siehe auch Abbildung 3.2) ein Plugin programmiert, das Daten aus dem Systemmodell verwenden kann, um Anforderungen mittels einer Simulation zu überprüfen. Die Ergebnisse werden dabei automatisch im Systemmodell aktualisiert. Das Plugin dient als Demonstrator, wie eine automatisierte Anforderungsüberprüfung durchgeführt werden kann.

4.2 Betrachtung des Einsatzgebietes

Die erfolgreiche Umsetzung eines MBSE Vorhabens bedarf einer genauen Analyse des Einsatzgebietes. Dabei ist einerseits auf den im jeweiligen Unternehmen vorherrschenden Produktentwicklungsprozess einzugehen, wie auch auf die Produkte, die entwickelt wer-

den. Die Betrachtung des Einsatzgebietes wurde im Zuge der Präsenzphase bei der Firma Fill durchgeführt. Weiters wurden durch Gespräche mit Mitarbeitern aus der Technik und dem Projektmanagement die relevanten Aspekte für einen MBSE Ansatz herausgearbeitet.
Zu Beginn werden die Produkte betrachtet, die von der Firma Fill entwickelt werden. Wie im Kapitel 1 erwähnt, handelt es sich bei den von Fill entwickelten und produzierten Produkten hauptsächlich um Sonderanfertigungen und Kleinserien. Als Beispiele können folgenden Produkte genannt werden:

- Bearbeitungszentren
- Gießereimaschinen
- Maschinen für die Skifertigung
- usw...

In Abbildung 4.1 ist ein Beispiel eines Bearbeitungszentrums abgebildet. Im konkreten Fall handelt es sich um eine Syncromill C.

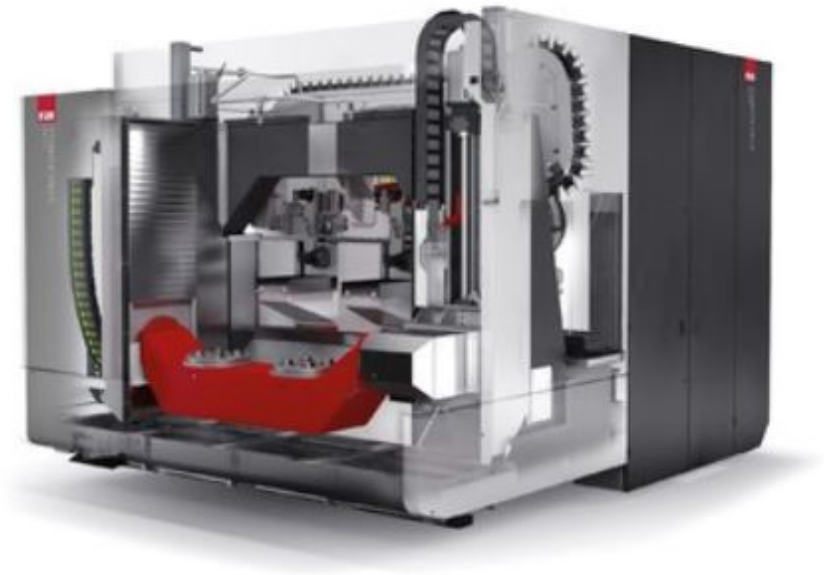


Abbildung 4.1: Beispielbild einer Syncromill nach [Fil]

Produkte der Firma Fill vereinen die drei Fachdisziplinen Mechanik, Eloktrotechnik und Software gleichermaßen. Demzufolge muss für diese Arbeit der verwendete MBSE Ansatz für mechatronische Produkte geeignet sein.
Aus wirtschaftlichen Gründen wird seitens der Firma Fill die Wiederverwendung von Modulen und Anlagenteilen angestrebt. Um dieses Vorhaben zu ermöglichen, muss das

ausgewählte Werkzeug zur Erstellung der Systemmodelle in der Lage sein, die erstellten Modelle zu kopieren und zu referenzieren.
Als zweiter Aspekt des Einsatzgebietes wurde der Produktentwicklungsprozess der Firma Fill betrachtet. Zur Beantwortung der Forschungsfragen sind nur die ersten Phasen des Produktentwicklungszyklus relevant. Aus diesem Grund werden auch nur die ersten Schritte des Produkentwicklungsprozesses der Firma Fill beschrieben.
Der Produktentwicklungsprozess gestaltet sich folgendermaßen. An erster Stelle steht bei jedem Projekt der Vertrieb. Dieser nimmt die Kundenwünsche auf und erstellt gemeinsam mit der Projektierung ein erstes Konzept des gewünschten Produktes. Im Anschluss daran werden erste Teile des Produktes konstruiert und simuliert. Bei dieser Simulation handelt es sich hauptsächlich um eine Anlagensimulation mit der Software *Visual Components*. Die durch die Simulation gewonnenen Erkenntnisse fließen in den weiteren Entwicklungsprozess ein. Die Elektrotechnik und die Softwaretechnik bilden die weiteren Schritte des Produktentwicklungsprozesses. Je nach Notwendigkeit wird die Simulation auch in den weiteren Phasen herangezogen, um frühzeitige Erkenntnisse zu gewinnen. Parallel dazu findet die Arbeitsvorbereitung statt. Im Anschluss daran wird mit der Fertigung begonnen. Die Phasen der Arbeitsvorbereitung und der Fertigung sind allerdings für diese Arbeit nicht relevant und werden daher nicht näher erläutert.
Durch die bewusste Betrachtung der Art der hergestellten Produkte, wird auf die individuellen Gegebenheiten Rücksicht genommen.

4.3 Auswahlprozess der Sprache, der Methode und des Werkzeugs

Unter Betrachtung des Einsatzgebietes und der Technik der Produkte ergeben sich für die Sprache, die Methode und das Werkzeug Anforderungen, die die gewählten Komponenten erfüllen müssen.

4.3.1 Anforderungen an die Sprache

Da die Firma Fill hauptsächlich im Sondermaschinenbau tätig ist, wird dieser Umstand bei den Anforderungen an die Sprache berücksichtigt. Demzufolge werden die Anforderungen an die Sprache in zwei Kategorien aufgeteilt. Eine Kategorie enthält die Anforderungen aufgrund der Technik der Produkte, die zweite enthält die Anforderungen aufgrund des Einsatzgebietes beziehungsweise des Produkentwicklungsprozesses der Firma Fill. In den folgenden Aufzählungen sind die Anforderungen dargestellt, auf deren Basis die Sprache ausgewählt wurde:

Anforderungen aufgrund der Technik der Produkte

1.1 Eignung für mechatronische Produkte

1.2 Die Anforderungen an das Produkt müssen abbildbar sein

1.3 Die Funktionen des Produktes müssen abbildbar sein

1.4 Die Komponenten des Produktes müssen abbildbar sein

1.5 Das Verhalten des Produktes muss abbildbar sein

1.6 Das Konzept der Objektorientierung muss möglich sein

1.7 Parameter der jeweiligen Komponenten müssen hinterlegbar sein

1.8 Die durch die Sprache generierten Informationen müssen sich für die Weiterverwendung im Produktentwicklungsprozess eignen

Anforderungen aufgrund des Einsatzgebietes

Quelle für diese Anforderungen sind einerseits Teile der Aufzählung aus dem Abschnitt 2.4.3, wo die Verwendung der Modelle im modellbasierten virtuellen Produktentwicklungsprozess dargestellt wird. Andererseits werden Anforderungen durch Gespräche mit Mitarbeitern der Firma Fill aus den Bereichen Technik und Projektmanagement und durch eigene Betrachtungen ermittelt. Als wichtige Erkenntnis daraus ergibt sich die Notwendigkeit, dass alle Projektbeteiligten in der Lage sein müssen, die Sprache zu verstehen, und auch verwenden können. Grund dafür ist der Umfang der Produkte der Firma Fill. Würden die Systemmodelle nur von wenigen Personen erstellt werden, kann der notwendige Detaillierungsgrad nicht erreicht werden, da der Aufwand zur Erstellung zu groß ist. Aus diesem Grund muss die gewählte Sprache für alle Beteiligten leicht zu erlernen und anzuwenden sein. Durch die Tätigkeit im Sondermaschinenbau, der von steigendem Zeitdruck geprägt ist, ist auch die Dauer der Erstellung eines Systemmodells nicht zu vernachlässigen. Weiters treten im Laufe des Entwicklungsprozesses ständig Anforderungsänderungen auf, die im Systemmodell aktualisiert werden müssen. Aus diesen Betrachtungen ergeben sich folgende Anforderungen.

2.1 Speichern von Informationen, die im Entwicklungsprozess anfallen

2.2 Gemeinsame Basis für Kommunikation über alle Fachdisziplinen hinweg

2.3 Quelle für Informationen für alle Projektbeteiligten

2.4 Erlangen eines gemeinsamen Systemverständnisses

2.5 Bereitstellung von Parametern des Systems

2.6 Kollaboratives Arbeiten muss möglich sein

2.7 Die Sprache muss von allen Projektbeteiligten verwendet werden können

2.8 Die Sprache muss schnell erlernt werden können

2.9 Die Sprache muss für ein Umfeld von ständigen Anforderungsänderungen geeignet sein

4.3.2 Auswahl der Sprache

Auf Basis der Anforderungen aus beiden Kategorien werden die Sprachen SysML und CONSENS verglichen und in Tabelle 4.1 gegenübergestellt.

Aus Tabelle 4.1 ist ersichtlich, dass bei der Betrachtung der technischen Anforderungen kein nennenswerter Unterschied zwischen den Sprachen ersichtlich ist. Anders verhält es sich hingegen bei der Betrachtung der Anforderungen, die aufgrund des Einsatzgebiets entstanden sind. Hier fällt auf, dass SysML die Anforderungen 2.2, 2.3 und 2.4 nicht oder nur teilweise erfüllt.
Auf Basis der Bewertung in Tabelle 4.1 wird in dieser Arbeit die Sprache CONSENS verwendet.

4.3.3 Anforderungen und Auswahl der Methode

Nach erfolgter Auswahl der Sprache wird eine geeignete Methode ausgewählt. Zur Auswahl der Methode werden zuerst die Betrachtungen des Einsatzgebietes miteinbezogen. Als zweiter Schritt wird die Vereinbarkeit der gewünschten Methode mit der vorher ausgewählten Sprache sichergestellt. Erst, wenn beide Aspekte als erfüllt angesehen werden können, kann die gewünschte Methode verwendet werden. Oberstes Ziel bei der Auswahl der Methode ist die Eignung für die Entwicklung mechatronischer Produkte. Ein weiteres Kriterium ist die Einfachheit der Umsetzung der gewählten Methode. Es soll vermieden werden, dass nur durch wiederholte Schulungen die Verwendung der Methode möglich ist. Aus diesem Grund wurde die Methode CONSENS in die engere Auswahl aufgenommen. Da ein Zusammenspiel der Methode CONSENS und der Sprache CONSENS gegeben ist, wird in dieser Arbeit diese Methode verwendet.

4.3.4 Anforderungen und Auswahl des Werkzeuges

Im letzten Schritt wird das Werkzeug ausgewählt. Die Auswahl verhält sich ähnlich zur Auswahl der Methode. Zuerst wird überprüft, ob das Werkzeug die Anforderungen aus der Betrachtung des Einsatzgebietes erfüllt. Im zweiten Schritt wird die Vereinbarkeit mit der bereits ausgewählten Sprache und Methode überprüft.
Am Ende des Auswahlprozesses erhält man ein Werkzeug, das für die gewünschte Sprache und Methode geeignet ist. Da nicht jedes Werkzeug in der Lage ist, jede Sprache und Methode abzubilden, ist die Auswahl in diesem Stadium des Auswahlprozesses schon eingeschränkt. Eine mögliche Option stellt die Software iQUAVIS dar. Zur Sicherheit wird die Erfüllung der Anforderungen an das Werkzeug dennoch überprüft.

Anforderungen an das Werkzeug

1.1 Verwendung der gewählten Sprache und Methode ermöglichen

1.2 Einfache Bedienung

1.3 Anbieten von Schnittstellen zur Weiterverwendung der Informationen aus dem Systemmodell

Nummer	Anforderung	CONSENS	SysML
1.1	Eignung für mechatronische Produkte	●	●
1.2	Die Anforderungen an das Produkt müssen abbildbar sein	●	●
1.3	Die Funktionen des Produktes müssen abbildbar sein	●	●
1.4	Die Komponenten des Produktes müssen abbildbar sein	●	●
1.5	Das Verhalten des Produktes muss abbildbar sein	●	●
1.6	Das Konzept der Objektorientierung muss möglich sein	●	●
1.7	Parameter der jeweiligen Komponenten müssen hinterlegbar sein	●	●
1.8	Die durch die Sprache generierten Informationen müssen sich für die Weiterverwendung im Produktentwicklungsprozess eignen	●	●
2.1	Speichern von Informationen die im Entwicklungsprozess anfallen	●	●
2.2	Gemeinsame Basis für Kommunikation über alle Fachdisziplinen hinweg	●	●
2.3	Quelle für Informationen für alle Projektbeteiligten	●	●
2.4	Erlangen eines gemeinsamen Systemverständnisses	●	●
2.5	Bereitstellung von Parametern des Systems	●	●
2.6	Kollaboratives Arbeiten muss möglich sein	●	●
2.7	Die Sprache muss von allen Projektbeteiligten verwendet werden können	●	◐
2.8	Die Sprache muss schnell erlernt werden können	●	○
2.9	Die Sprache muss für ein Umfeld von ständigen Anforderungsänderungen geeignet sein	●	◐

Tabelle 4.1: Gegenüberstellung der Sprachen unter Berücksichtigung der Anforderungen der Firma Fill

1.4 Kopieren und Referenzieren von bestehenden Modellen ermöglichen

1.5 Kollaboratives Arbeiten ermöglichen

1.6 Übersichtliche Darstellungen ermöglichen

1.7 Rechtevergabe ermöglichen

Von der Software iQUAVIS können alle Anforderungen erfüllt werden. Aus diesem Grund werden in dieser Arbeit die Systemmodelle mit iQUAVIS erstellt. Der Vollständigkeit halber gilt es zu erwähnen, dass iQUAVIS Funktionen im Bereich der Erstellung des Systemmodells bietet, als auch Funktionen für das Projektmanagement. Dazu zählen etwa das Erstellen von Gantt-Charts oder das Zuweisen von Aufträgen an MitarbeiterInnen. Das Hauptaugenmerk in dieser Arbeit liegt allerdings auf der Erstellung von Systemmodellen mit iQUAVIS.

4.4 Erstellen der Modelle

Nach erfolgreicher Auswahl der Sprache, der Methode und des Werkzeuges werden zwei Modelle auf Basis bereits existierender Fill Produkte erstellt.
Im ersten Modell wird eine Syncromill C21-63/600 als Modell abgebildet. Das zweite Modell stellt einen Teil einer verketteten Anlage dar. Dieser Teil der Anlage wird in einer Anlagensimulation simuliert, um die Verknüpfung von Simulation und Systemmodell näher betrachten zu können. Das Erstellen der Modelle soll einerseits dazu dienen, die Eignung der ausgewählten Komponenten zu testen, andererseits bildet das zweite Modell die Ausgangsbasis für die Verknüpfung der Informationen aus dem Systemmodell mit den nachfolgenden Phasen des Entwicklungsprozesses.

4.5 Verknüpfen des Systemmodells

Die Verknüpfung des Systemmodells wird anhand der automatisierten Anforderungsüberprüfung verdeutlicht. Ziel hierbei ist es, Anforderungen, welche im Systemmodell definiert werden, mit Hilfe einer Simulation automatisiert zu überprüfen und das Ergebnis im Systemmodell darzustellen. Die Simulation und das Simulationsprogramm sind seitens der Firma Fill vorgegeben. Die Visualisierung der Anforderungen und der Ergebnisse erfolgt einerseits im Systemmodell und andererseits mit Hilfe eines Plugins im Simulationsprogramm. Grundvoraussetzung sind geeignete Schnittstellen, die ein Vorhaben dieser Art zulassen. Verwendet wird dazu die *.Net* -Schnittstelle des Simulationsprogrammes und die REST-API des Werkzeugs, mit dem das Systemmodell erstellt wird.

4.6 Erstellung der Systemmodelle als Basis eines Digitalen Zwillings im Entwicklungsprozess

Damit die Informationen eines Systemmodells im weiteren Entwicklungsprozess verwendet werden können, müssen die Informationen des Systems in geeigneter Form erstellt und gespeichert werden.

4.6.1 Grundlegendes zum Werkzeug iQUAVIS

iQUAVIS bietet die Möglichkeit Systemmodelle mit Hilfe von CONSENS zu erstellen. Zur Darstellung der Systemmodelle stehen folgende Möglichkeiten zur Verfügung:

- Baumstruktur
- Diagramme
- Tabellen
- Zeitpläne
- Matrizen

Weiters können die Informationen des Systemmodells in verschiedenen Datentypen gespeichert werden. Standardmäßig gibt es vier verschiedene Datentypen. Die Auswahl kann allerdings durch benutzerdefinierte Klassen unbegrenzt erweitert werden, was es dem Ersteller des Modells erlaubt, Klassen je nach Notwendigkeit anzulegen. Die vier grundlegenden Datentypen sind folgende:

- Element
- Anforderung
- Aufgabe
- Angelegenheit[1]

Je nach Darstellungsvariante werden die Objekte der Datentypen in einer anderen Art und Weise dargestellt. iQUAVIS bietet jedoch in den meisten Bereichen die Möglichkeit, benutzerdefinierte Einstellungen vorzunehmen, was dazu führt, dass das Werkzeug gut an das Einsatzgebiet angepasst werden kann. Auf den nächsten Seiten werden die Darstellungsvarianten und die Datentypen näher erläutert. Im Anschluss daran werden die erstellten Systemmodelle erklärt.

Baumstruktur
Die Baumstruktur erlaubt die Darstellung von Informationen und deren Zusammenhänge

[1] Der Datentyp Angelegenheit wird hauptsächlich im Zusammenhang einer Fehlermöglichkeits- und Einflussanalyse (FMEA) verwendet, kann aber auch für allgemeinere Zwecke verwendet werden.

aus dem Systemmodell. In Abbildung 4.2 ist ein Beispiel einer Baumstruktur zu sehen. In dieser Abbildung werden die Anwendungsszenarien, die Anforderungen, die Funktionen und die Elemente dargestellt. Die grauen Objekte im linken Bereich der Abbildung 4.2 stellen die Anwendungsszenarien dar. Die nächste Gruppe von Objekten sind die Anforderungen und Funktionen, diese werden von den grünen Objekten dargestellt. Auf der rechten Seite der Abbildung werden die Elemente mit blauen Objekten verdeutlicht. Die blauen Linien stellen die Zusammenhänge zwischen den Objekten dar. Welche Informationen in einer Baumstruktur auftauchen sollen, kann vom Benutzer frei gewählt werden. Besteht beispielsweise der Wunsch nur Anforderungen und Funktionen darzustellen, kann eine neue benutzdefinierte Baumstruktur angelegt werden, die stets die aktuellen Anforderungen und Funktionen darstellt.

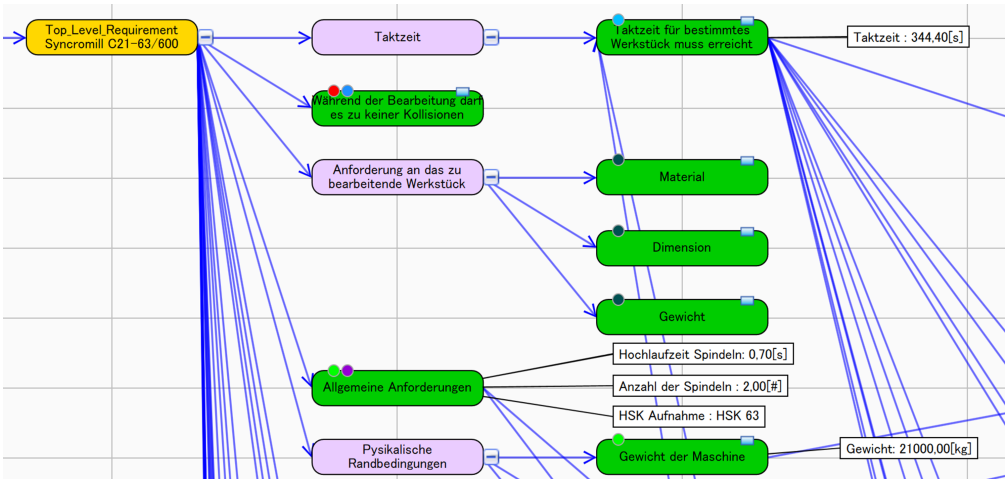


Abbildung 4.2: Beispiel einer Baumstruktur

Benutzdefinierte Baumstrukturen ermöglichen eine übersichtliche Darstellung der Zusammenhänge. Weiters besteht die Möglichkeit in einer Baumstruktur Filter anzuwenden, um mehr Übersichtlichkeit zu generieren. In Abbildung 4.3 ist eine mögliche Auswahl an Baumstrukturen zu sehen.

Diagramme

iQUAVIS verwendet vier verschiedene Diagrammtypen:

- Zustandsdiagramm
- Funktionsblockdiagramm
- Elementblockdiagramm
- Sequenzdiagramm

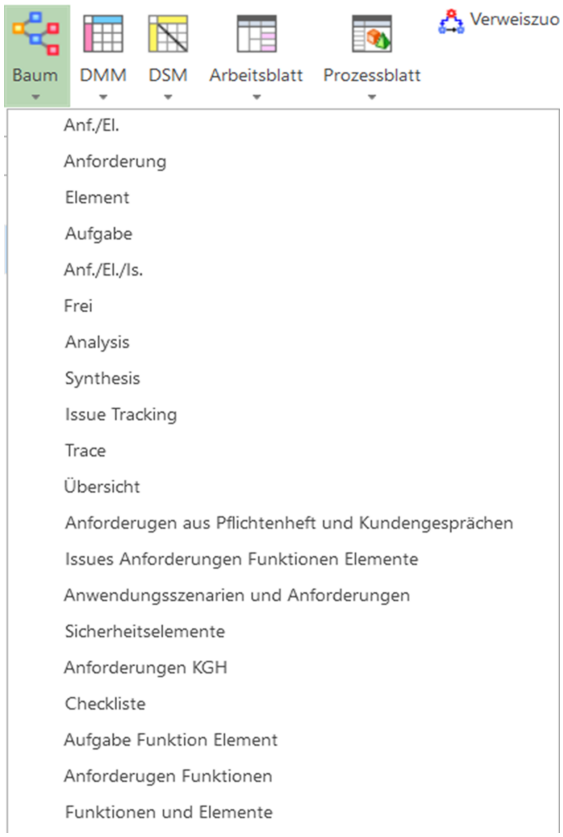


Abbildung 4.3: Verschiedene Baumstrukturen

Tabellen
Jegliche Information aus dem Systemmodell kann in Form von Tabellen dargestellt werden. Welche Inhalte in der Tabelle dargestellt werden, kann vom Benutzer festgelegt werden.
Da es sich bei den Tabellen ebenfalls um eine Darstellungsform der Informationen handelt, werden Änderungen die in den Tabellen vorgenommen werden, im Systemmodell gespeichert.

Datentypen
Die Erstellung von neuen Objekten kann in allen Darstellungsformen stattfinden. In der Praxis werden hauptsächlich Diagramme und die Baumstruktur dafür verwendet. Über Eingabemasken werden die Informationen des Objekts hinterlegt. Beipielsweise werden die zugehörige Klasse oder die Paramter des Objekts festgelegt. Weiters werden in diesem Schritt auch die Links zu den zugehörigen Dokumenten und Datenbanken hinterlegt. Jedem erstellten Objekt wird eine eindeutige ID zugewiesen, die es ermöglicht, das Objekt auch bei der Verwendung von Schnittstellen zu identifizieren. In Abbildung 4.4 ist die Eingabemaske für ein Systemelement dargestellt.

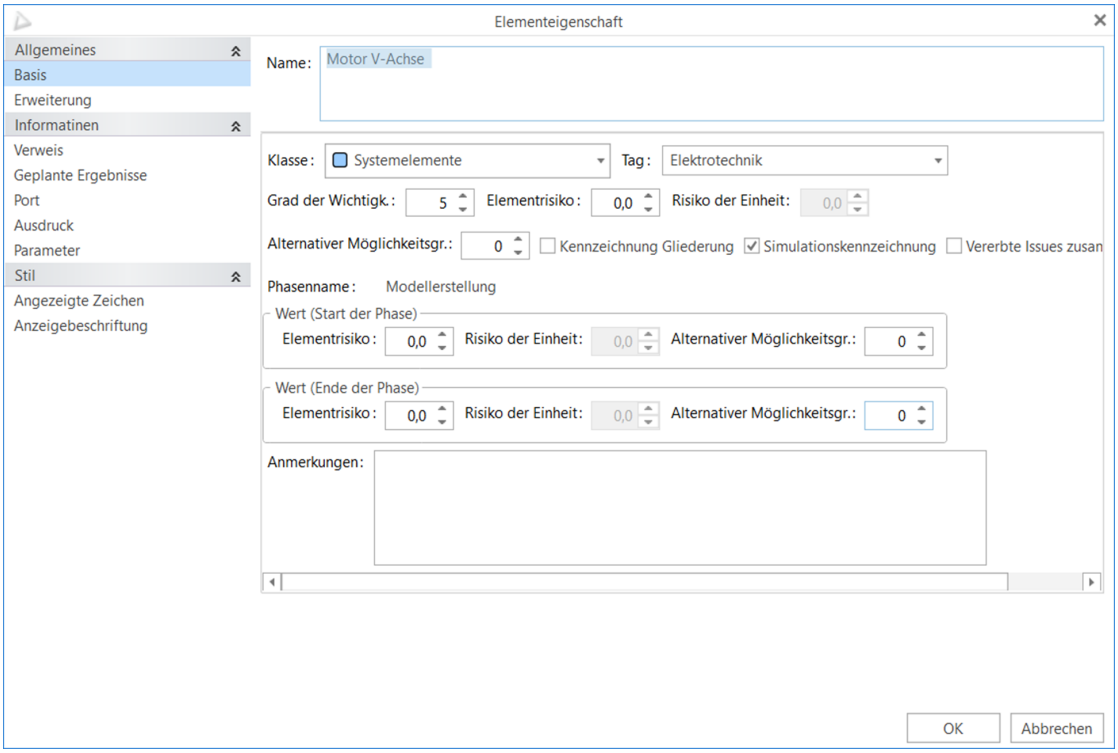


Abbildung 4.4: Eingabemaske Systemelement am Beispiel eines Motors

Dasselbe Prinzip wird bei der Erstellung von Objekten anderer Datentypen angewandt. Jedes Objekt eines jeden Datentyps kann mit Tags versehen werden. Es kann frei gewählt werden, welche Tags zu vergeben sind. Bei großen Systemen ist die systematische Verwendung von Tags hilfreich, da nach diesen gefiltert werden kann.

4.6.2 Aufbau des Systemmodells der Syncromill

Zum Aufbau des Systemmodells wird die Methode CONSENS aus Abschnitt 2.3.5 verwendet. Zur Erstellung eines vollständigen Systemmodells müssen die sechs Schritte der Methode mehrmals durchlaufen werden. Ein weiterer Grundsatz ist die Modellierung vom Groben ins Detail. Um die Navigation durch das Systemmodell zu erleichtern, werden Objekte mit zugehörigen Ansichten verknüpft. In den meisten Fällen führt die Verknüpfung zu einer detaillierteren Ansicht des Objektes.
Das folgende Systemmodell bildet ein Bearbeitungszentrum der Firma Fill ab.

Bearbeitungszentrum Syncromill C - Umfeldmodell
Der erste Schritt der CONSENS Methode betrachtet das Umfeld des Systems. Bei der Syncromill weisen beispielsweise die Energieversorgung oder die Betriebsmittelversorgung Schnittstellen mit der Umwelt auf. In Abbildung 4.5 ist das Umfeldmodell der Syncromill

in iQUAVIS dargestellt.

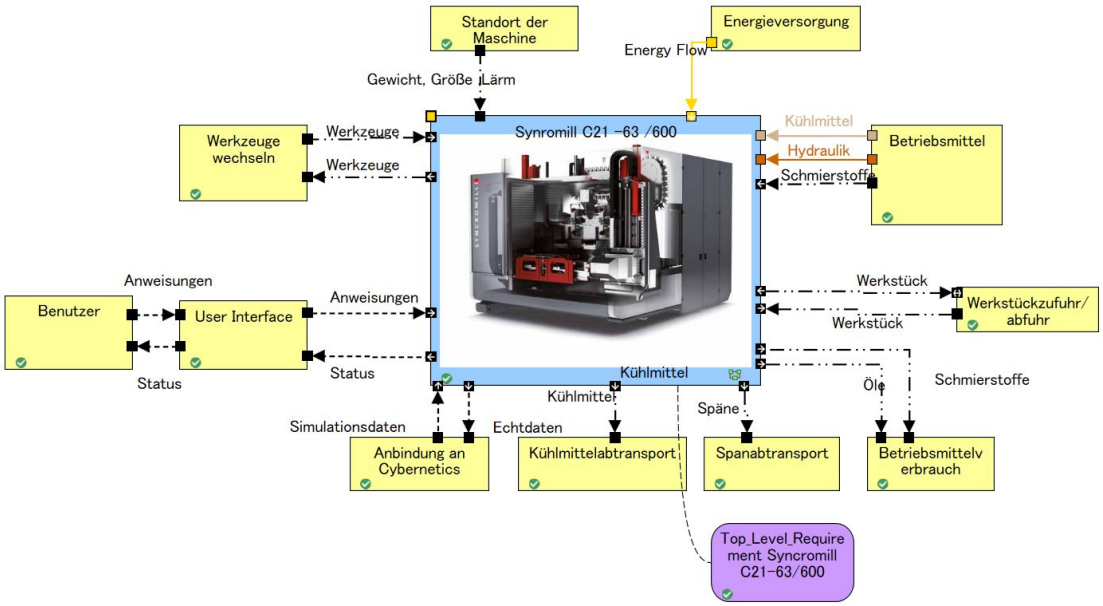


Abbildung 4.5: Umfelddiagramm Syncromill

Die Umweltelemente werden durch die gelben Rechtecke dargestellt. Das Modell der Syncromill ist in Abbildung 4.5 als Blackbox ausgeführt. Erst durch die Betrachtung der Inhalte dieser Blackbox werden die Baugruppen der Syncromill sichtbar. Dieses Blackbox-Prinzip führt laut Haberfellner u. a. zur Reduktion von Komplexität. Aus diesem Grund wird dieses Prinzip im gesamten Systemmodell angewendet.
Schnittstellen werden wie in Abbildung 4.5 mit Verbindungspfeilen dargestellt. Diese können benutzerdefiniert erstellt und zugewiesen werden.
Mit Hilfe der Umfeldbetrachtung der Syncromill soll für alle Beteiligten dargestellt werden, welche Interaktionen mit der Umwelt auftreten. Diese Interaktionen können direkten Einfluss auf die Syncromill haben. Beispielsweise gibt es je nach Staat unterschiedliche gesetzliche Bestimmungen, die ein Bearbeitungszentrum erfüllen muss. Ein weiteres Beispiel für eine Interaktion mit der Umwelt ist der Spanabtransport. Wie dieser ausgeführt wird, hängt von den vorherrschenden Gegebenheiten beim Kunden ab.
In Abbildung 4.5 sind nur die Namen der Umweltelemente zu sehen. Allerdings können den Umweltelementen, wie auch allen anderen Objekten im Systemmodell, mehr Informationen zugewiesen werden. In Abbildung 4.4 ist ersichtlich, welche Informationen einem Umweltelement zugewiesen werden können. Da Systemelemente und Umweltelemente zum selben Datentyp gehören, haben beide Elemente dieselbe Eingabemaske. Zu den zusätzlich hinterlegbaren Informationen gehören:

- Verweis - Ein Verweis ist ein Link zu einem Dokument, das genauere Informationen über das betreffende Objekt enthält. Im Fall des Umweltelements *Spanabtransport*

ist das ein Dokument, das die Beschreibung der Gegebenheiten beim Kunden enthält.

- Parameter - Die Hinterlegung von Parametern ermöglicht die Beschreibung des Objekts mit entsprechenden Größen. Am Beispiel des Umweltelements *Energieversorgung* aus Abbildung 4.4 ist ein Parameter die Versorgungsleistung.
- Erweiterung - Neben den vordefinierten Informationen können benutzerdefinierte Informationen hinzugefügt werden. Dadurch ist es möglich, dass jede Art von Information in einem Objekt gespeichert werden kann.

Nach Erstellung des Umfeldmodells folgt laut der CONSENS Methode die Betrachtung der Anwendungsszenarien.

Bearbeitungszentrum Syncromill C - Anwendungsszenarien
In Abbildung 4.6 sind die Anwendungsszenarien der Syncromill dargestellt. Das Anwendungsszenario *Bearbeitungsbetrieb* ist mit einem Funktionsdiagramm verknüpft. Diese Verknüpfung erlaubt die einfache Navigation zu dem dazugehörigen Funktionsdiagramm.

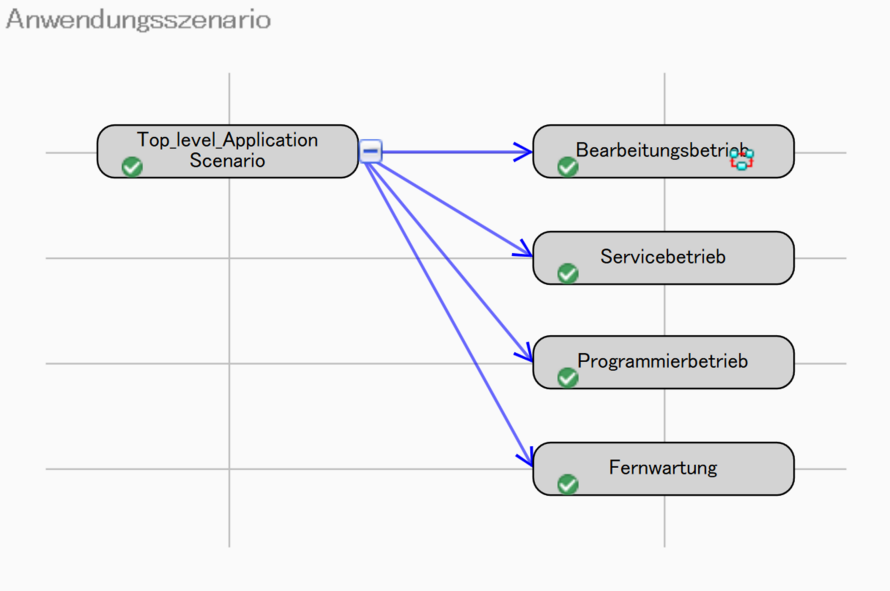


Abbildung 4.6: Anwendungsszenarien Syncromill

Neben dem Bearbeitungsbetrieb gibt es noch weitere Anwendungsszenarien. Dazu zählen der Servicebetrieb, der Programmierbetrieb und die Fernwartung. Hauptgrund für die Darstellung der Anwendungsszenarien ist der nächste Schritt in der CONSENS Methode, die Anforderungen. Durch die Darstellung, welchen Anwendungsszenarien das System gerecht werden muss, ergeben sich entsprechende Anforderungen.

Bearbeitungszentrum Syncromill C - Anforderungen
Zur Darstellung der Anforderungen eines Systems wird die Baumstruktur verwendet. Durch die große Anzahl an Anforderungen, die die Syncromill aufweist, ist eine Gliederung notwendig. Demzufolge sind die Anforderungen, wenn möglich, den entsprechenden Baugruppen zugeordnet. In Abbildung 4.7 ist ein Ausschnitt aus den Anforderungen der Syncromill zu sehen. Um die Gliederung besser zu veranschaulichen, sind die Bottom-Level Anforderungen in grün eingefärbt. Bei der Erstellung von Anforderungen werden die in Abschnitt 2.1 angeführten Attribute festgelegt. In iQUAVIS wird jedem Objekt bei dessen Erstellung eine im Systemmodell eindeutige ID zugewiesen. Diese ID kann zur Identifizierung der Anforderung herangezogen werden. Die automatisch erstellte ID besteht aus einer Kombination aus Zahlen und Buchstaben und hat nicht das typische Format einer Anforderungs-ID (z.B: 5.1.2). Bei Bedarf kann eine solche mit Hilfe der benutzerdefinierten Attribute hinzugefügt werden. Eine automatische Überprüfung auf doppelte Belegung der ID ist dabei allerdings nicht gegeben. Bei den Anforderungen der Syncromill werden den Bottom-Level Anforderungen nur mehr die Beschreibungen zugewiesen. Namen werden bei den darüberliegenden Ebenen verwendet. Bei Bedarf kann allerdings jede Anforderung mit Namen und Beschreibungen versehen werden. Der Status der Anforderungen kann über zu vergebende Tags dargestellt werden. Welche Tags zu vergeben sind, kann frei gewählt werden. Eine Information über die Version der Anforderungen ist standardmäßig nicht vorgesehen, kann aber durch benutzerdefinierte Attribute hinzugefügt werden. iQUAVIS bietet allerdings die Möglichkeit, die Historie des Systemmodells darstellen zu können. In dieser sind die verschiedenen Versionen sichtbar. Die in Abbildung 4.7 dargestellten Anforderungen betreffen den Werkzeugträger der Syncromill. Eine Beispielanforderung ist die „Chip-to-chip“ Zeit. Diese Zeitspanne beschreibt die Dauer vom Spanabtrag über den Werkzeugwechsel zum erneuten Spanabtrag. Die Beschreibung dieser Anforderung lautet: *Chip to Chip Zeit darf 2.7s nicht überschreiten.* Weiters wird der Anforderungen der Parameter *chip_ to_ chip_ time* mit 2,7 Sekunden hinzugefügt. Über eine Auswahl von Tags kann der Status angezeigt werden.
Neben der Statusanzeige wird mit den Tags eine weitere Gliederung vorgenommen. Die erste Gliederung erfolgt über die Unterteilung in die Baugruppen, die zweite über die Markierung mit welchen Simulationsprogrammen die Anforderungen überprüft werden können. Demzufolge werden Tags für alle Simulationsprogramme, die im Entwicklungsprozess bei Fill eingesetzt werden, erstellt und den jeweiligen Anforderungen zugewiesen. Die nachfolgende Aufzählung stellt einen Auszug aus den zu vergebenden Tags im Systemmodell der Syncromill dar:

- Wichtig
- Unbesprochen
- FIX
- Erfüllt
- Matlab/FEM/MKS_Anforderung

- VC_Anforderung (VC - Visual Components)
- CAD_Anforderung
- Datenblatt_Anforderung
- Cut_View_Anforderung
- ESPRIT_Anforderung

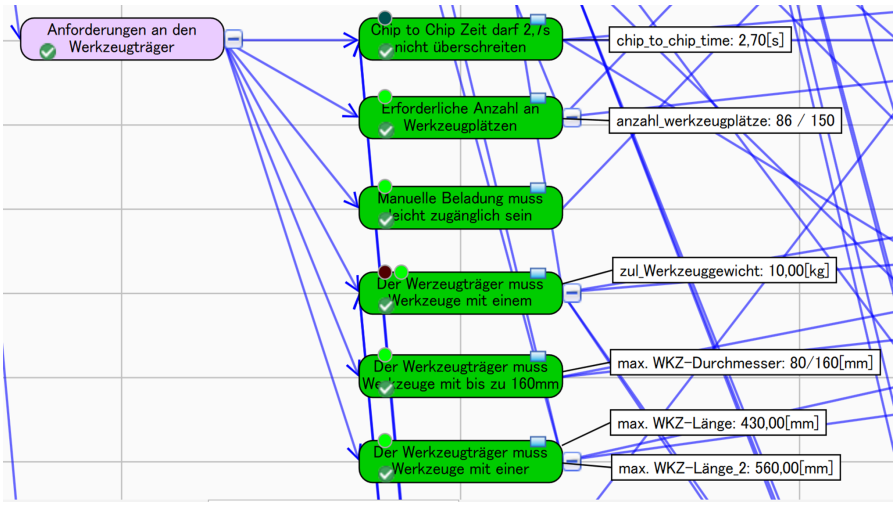


Abbildung 4.7: Anforderungen an den Werkzeugträger der Syncromill

Je nach Art der möglichen Überprüfung werden die Tags den Anforderungen zugewiesen. Durch diese Funktion ist es möglich, bei großen und unübersichtlichen Systemmodellen die Übersicht zu behalten.

Nach der Erstellung der Anforderungen, werden aus diesen die Funktionen abgeleitet. Die Funktionen bilden den vierten Schritt der CONSENS Methode.

Bearbeitungszentrum Syncromill C - Funktionen
Funktionen lassen sich großteils aus den erstellten Anforderungen ableiten. Die Vorgehensweise dabei ist, sich die Frage zu stellen, welche Funktionen notwendig sind, um eine bestimmte Anforderung zu erfüllen. Die Funktionen können in einer Baumstruktur oder in einem Funktionsdiagramm definiert werden. Da sich Funktionen von Anforderungen ableiten, bietet sich eine Ansicht an, in der Anforderungen und Funktionen gemeinsam dargestellt sind. In Abbildung 4.8 werden die Funktionen der Werkzeugwechseleinheit dargestellt. Die von links kommenden blauen Pfeile stellen die Verbindung mit den Anforderungen dar. Die nach rechts wegzeigenden Pfeile stellen die Verbindung mit den Systemelementen dar. Damit kann in einer Ansicht die Verbindung von Anforderung, Funktion und Systemelement dargestellt werden.

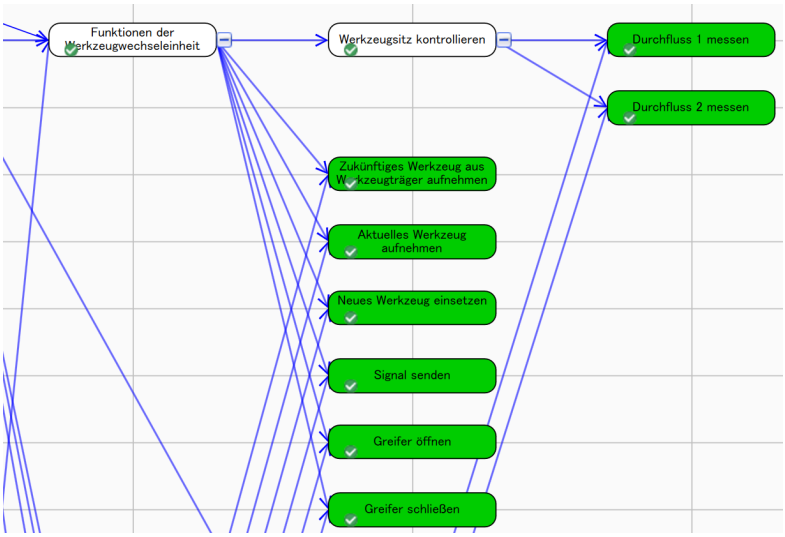


Abbildung 4.8: Funktionen der Werkzeugwechseleinheit der Syncromill

Um die Übersicht zu bewahren, erfolgt die Gliederung der Funktionen ebenfalls nach den Baugruppen der Syncromill. Bei der Formulierung der Funktionen wird darauf geachtet, mehr als ein Nomen und ein Verb zu verwenden. Bei der Verwendung lediglich eines Nomens und eines Verbs pro Funktion, besteht das Problem, dass diese Formulierung zu Verwirrungen führen kann. Die mögliche Verwirrung kann am Beispiel der Funktion: *Zukünftiges Werkzeug aus Werkzeugträger aufnehmen* gezeigt werden. Bei der Verwendung von lediglich eines Nomens und eines Verbs, würde die Funktion *Werkzeug aufnehmen* lauten. In diesem Fall ist aber nicht klar, ob das Werkzeug aus dem Werkzeugträger, oder aus der Spindel aufgenommen werden soll. Zur Vermeidung solcher Missverständnisse werden die Funktionen mit dem Ort in Verbindung gebracht, an dem diese ausgeführt werden. Dieser Schritt erleichtert die Zuordnung der Funktion, wenn diese im Systemmodell einzeln und nicht im Zusammenhang mit der Baugruppe verwendet werden.
Funktionen werden in der frühen Phase des Entwicklungsprozesses erstellt. Im Bezug auf den Produktentwicklungsprozess der Firma Fill ist es daher möglich, die Entwicklung teilweise zu parallelisieren. Ohne Funktionsbetrachtung muss die Konstruktion abgeschlossen sein, damit die Abteilung Elektrotechnik die Arbeiten aufnehmen kann. Ebenso verhält es sich zwischen der Elektrotechnik und der Softwareabteilung. Erst bei abgeschlossener Arbeit der Elektrotechnik, kann die Softwareabteilung die Arbeit aufnehmen. Durch die funktionsorientierte Betrachtung sind die nach der Konstruktion folgenden Abteilungen in der Lage, Architekturen zu erstellen und gegebenenfalls schon bestehende Funktionseinheiten zu übernehmen. Die Detailausführungen werden durchgeführt, sobald die notwendigen Informationen vorliegen. Die Parallelisierung hat das Potential die Entwicklungszeit zu verkürzen und die Elektrotechnik und Softwareabteilung schon früh im Entwicklungsprozess einzubinden.
Nach erfolgreich erstellter Funktionsstruktur wird die Wirkstruktur erstellt.

Bearbeitungszentrum Syncromill C - Wirkstruktur
Erst im fünften Schritt der CONSENS Methode wird die technische Umsetzung der Anforderungen beziehungsweise der Funktionen betrachtet. Um die Informationen aus der technischen Umsetzung im Systemmodell zu speichern, werden Systemelemente für alle Komponenten eines Systems verwendet. Bei Bedarf besteht die Möglichkeit weitere Klassen einzuführen. Der Diagrammtyp, der zur Darstellung der Systemelemente verwendet wird, heißt Elementblockdiagramm.
Bei der Erstellung der Wirkstruktur wird das Vorgehen nach dem Prinzip *Vom Groben ins Datail* am häufigsten verwendet. Ausgehend vom Umfeldmodell in Abbildung 4.5, in dem die gesamte Syncromill als Blackbox dargestellt ist, wird auf den nächsten Ebenen der Detaillierungsgrad stetig erhöht. Die Detaillierung wird soweit fortgeführt, bis alle technisch relevanten Komponenten im Systemmodell auftauchen.
Am Beispiel der Syncromill gibt es folgende Detaillierungsebenen:

1 Umfeldmodell
Das Umfeldmodell stellt mit der Syncromill als Blackbox die oberste Ebene dar.

2 Baugruppendiagramm
Eine Ebene unter dem Umfeldmodell befindet sich das Baugruppendiagramm. Dieses enthält alle Baugruppen der Syncromill. Zu diesen Baugruppen zählen:
   - Werkzeugwechseleinheit
   - Schaltschrank
   - Hauptachsensystem
   - Pneumatik-Steuereinheit
   - Kompressor
   - Werkzeugträger
   - Zentrale Steuereinheit
   - Grundgestell
   - Sicherheitsbauteile
   - Beladetüren
   - User Interface
   - Verkleidung
   - Spanabtransport
   - Hydraulikanlage
   - Werkzeugwechselschott
   - Kühlmittelanalge
   - Schmiermittelanlage
   - Rundtischkombination

3 Detaillierung der Baugruppen
Baugruppen aus dem Baugruppendiagramm werden in einem neuen Diagramm weiter detailliert.

Zur Eingabe von Informationen eines Systemelements wird die Eingabemaske aus Abbildung 4.4 verwendet. Als Verweis werden bei Systemelementen Links zu den CAD-Daten, E-Plan Daten, Datenblättern oder Berichten hinterlegt. Bei richtiger Verlinkung führt der hinterlegte Link immer zur aktuellsten Version. Als Parameter werden Abmessungen, Massen, Leistungsangaben sowie sämtliche weitere relevante Werte hinterlegt.
Damit das Systemmodell Aufschluss über die Interaktion der Systemelemente geben kann, werden Funktionslinien hinzugefügt. Neben den vordefinierten Typen von Funktionslinien werden für die Syncromill benutzerdefinierte Funktionslinien angelegt. Bei dem Systemmodell der Syncromill werden folgende Funktionslinien verwendet:

- Elektrischer Strom
- Pneumatik
- Hydraulik
- Steuerungssignal
- Sensorsignal
- Erdung
- Kühlmittel
- Schmierstoffe

In Abbildung 4.9 sind Funktionslinien und die dazugehörigen Systemelemente abgebildet. Der Übergang von Funktionslinien zu Systemelementen wird durch *Ports* gebildet. In den Ports wird festgelegt, was über die Grenzen des Systemelements transportiert wird. Welche Art von Port verwendet wird, wird bei der Modellierung automatisch über die dazugehörige Funktionslinie festgelegt. Bei einer funktionierenden Verknüpfung sind Port und Funktionslinie vom selben Typ. Sind Port und Funktionslinie nicht vom selben Typ, wird diese Verbindung im Elementblockdiagramm als fehlerhaft dargestellt. Da die Funktionslinie immer mit einem Port vom selben Typ verbunden sein muss, entsprechen die verfügbaren Port-Typen genau den Typen der Funktionslinien.
Durch die Kombination von Systemelementen und Funktionslinien entsteht eine Sicht auf das Systemmodell, die Aufschluss über den Aufbau der Baugruppen gibt. In Abbildung 4.9 ist der Aufbau der Werkzeugwechseleinheit und eine dazugehörige Achse im Detail dargestellt.
Nach dem Vorgehen vom MVPE-Modell in Abbildung 2.33 werden die Systemkomponenten in zwei Schritten erarbeitet. Der erste Schritt ist die Darstellung der logischen Elemente, im zweiten Schritt werden die physikalischen Elemente abgeleitet. Die in Abbildung 4.9 dargestellten Systemelemente stehen für die logischen Elemente. Zur Beschreibung der physikalischen Elemente werden über die Eingabemaske die notwendigen

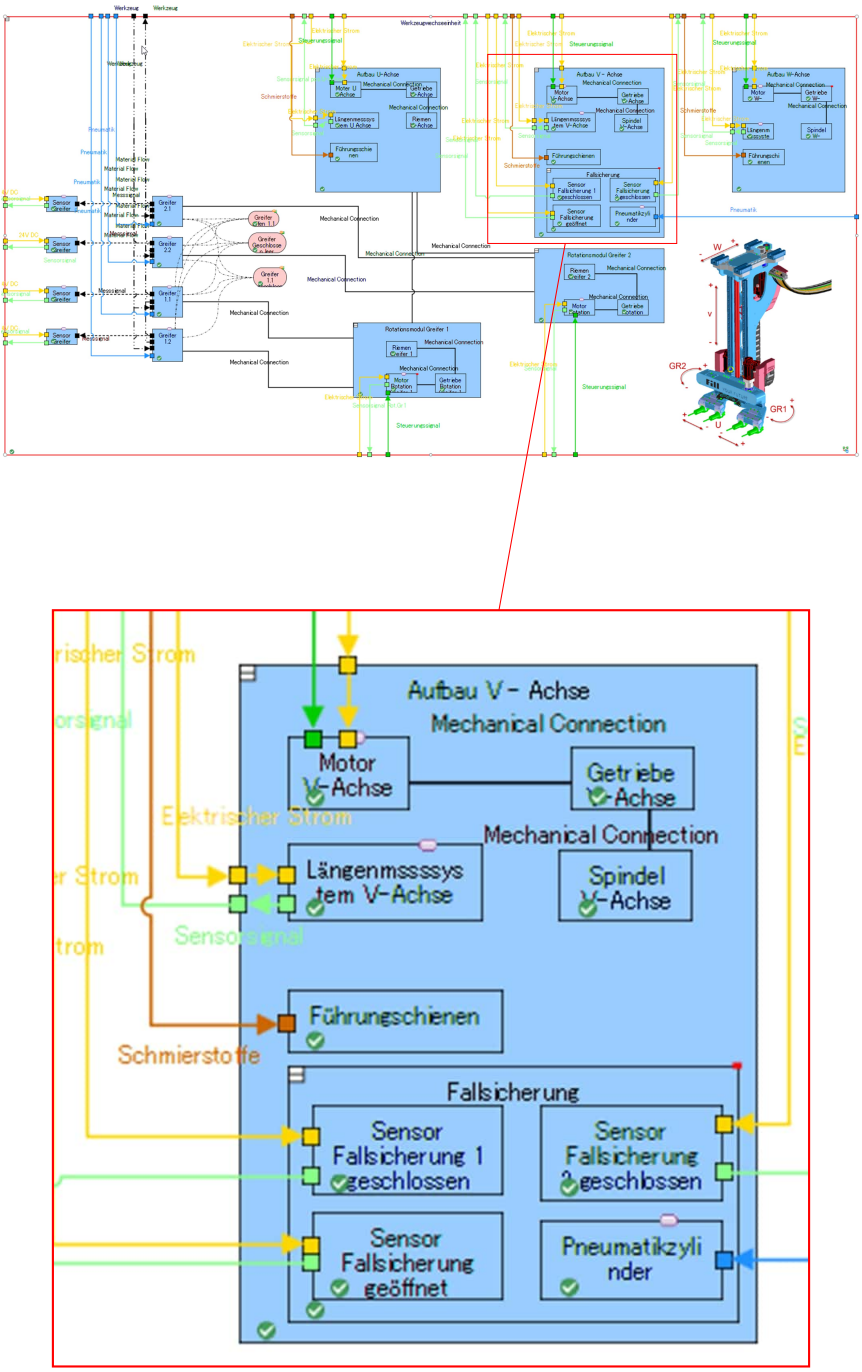


Abbildung 4.9: Wirkstruktur Werkzeugwechseleinheit Syncromill im Elementblockdiagramm

Details hinzugefügt, um die Komponente zweifelsfrei identifizieren zu können.

Nach der Erstellung der Wirkstruktur mit Hilfe der Elementblockdiagramme wird im sechsten Schritt der CONSENS Methode das Verhalten des Systems modelliert.

Bearbeitungszentrum Syncromill C - Verhalten

Zur Modellierung des Verhaltens wird auf die Funktionen zurückgegriffen, die im vierten Schritt der CONSENS Methode erstellt werden. Damit das Verhalten der Syncromill mit Hilfe der bereits bestehenden Funktionen abgebildet werden kann, werden Funktionsablaufdiagramme und Zustandsdiagramme verwendet. In Abbildung 4.10 ist das Funktionsablaufdiagramm der Werkzeugwechseleinheit abgebildet.

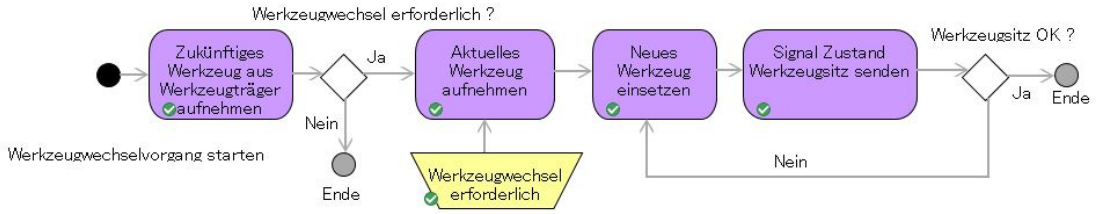


Abbildung 4.10: Funktionsablaufdiagramm einer Werkzeugwechseleinheit (ohne Bewegung der Greifer)

Bei den in Abbildung 4.8 und in Abbildung 4.10 dargestellten Funktionen, handelt es sich um dieselben Daten aus dem Systemmodell. Dieses Verhalten ist ein Beispiel für die Unterscheidung von Sicht und Modell (Abbildung 2.14) bei der Modellierung von Systemen. Weiters bedeutet das, dass Änderungen die in einer Ansicht durchgeführt werden, automatisch in allen anderen Ansichten übernommen werden.
Neben den Funktionsablaufdiagrammen werden auch Zustandsdiagramme verwendet, um das Verhalten eines Systems darzustellen. In Abbildung 4.11 ist das Zustandsdiagramm für den Werkzeugwechsel abgebildet. Um darzustellen, welche Funktion nötig ist, um einen bestimmen Zustand zu erreichen, werden die dazugehörigen Funktionen in das Zustandsdiagramm hinzugefügt. Die hinzugefügten Funktionen liegen auch in diesem Fall schon im Systemmodell vor und müssen nicht mehr erstellt werden. Auch hier kommt das Konzept der Trennung von Sicht und Modell zum Einsatz.
Mit der Modellierung des Verhaltens der Syncromill ist die CONSENS Methode vollständig durchlaufen. Dieser Prozess wird für die folgenden Baugruppen durchgeführt:

- Werkzeugwechseleinheit
- Hauptachsensystem
- Sicherheitsbauteile
- Spanabtransport
- Beladetüren

- Hydraulikanlage
- Kühlmittelanlage
- Rundtischkombination

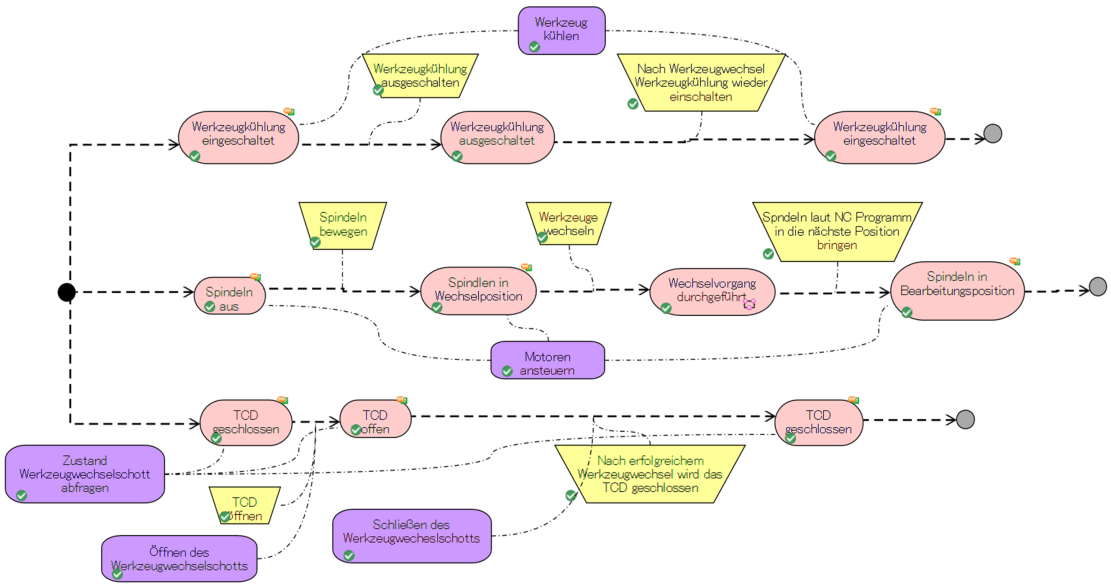


Abbildung 4.11: Zustandsdiagramm Werkzeugwechsel der Syncromill

Zusammenfassend enthält das Systemmodell zu diesem Zeitpunkt folgende Informationen:

- Informationen über das Umfeld der Syncromill:
  Dazu zählen Datenblätter, Informationen über die Energieversorgung, Betriebsmittelfluss, Länderbestimmungen, zulässige Abmessungen und Massen, Links zu anderen Datenbanken usw.
- Informationen über die Anwendungsszenarien der Syncromill:
  Neben der Auflistung der Anwendungsszenarien in der Baumstruktur liegen weiterführende Informationen in Form von Dokumenten oder Links zu Datenbanken vor.
- Informationen über die Anforderungen an die Syncromill:
  Die Anforderungen liegen mit den dazugehörigen Attributen in einem Datenformat vor, auf das über Schnittstellen zugegriffen werden kann. Das ermöglicht die zentrale Verwaltung von Anforderungen in iQUAVIS, mit der Möglichkeit die Anforderungen im weiteren Entwicklungsprozess ohne Medienbruch zu verwenden.
- Informationen über die Funktionen der Syncromill:
  Funktionen können ebenfalls mit Verweisen, Parametern und benutzerdefinierten

Informationen versehen werden. Die wichtigste Information der Funktionen ist eine aussagekräftige Bezeichnung und bei Bedarf eine Zuweisung zu einem Bauteil oder einer Baugruppe. Die weitere Verwendung der Funktionen im Entwicklungsprozess ohne Medienbruch ist hier ebenfalls gegeben.

- Informationen über die Wirkstruktur der Syncromill:
  In diesem Schritt der Modellierung werden die meisten Informationen hinterlegt. Einen Teil der Information liefert die aufgebaute Struktur, der Rest folgt durch Hinterlegung von Dokumenten, Verlinkung zu Datenbanken oder Vergabe von Parametern. Am Beispiel der Syncromill werden die Systemelemente zu den dazugehörigen CAD-Daten verlinkt. Diese Verlinkung führt dazu, dass die Projektbeteiligten ohne aktive Suche in einer CAD Datenbank über das Systemmodell Zugriff auf weiterführende Information erhalten. Weiters besteht auch hier die Möglichkeit der Verwendung der Systemelemente über Schnittstellen hinweg.

- Informationen über das Verhalten der Syncromill:
  Die Modellierung des Verhaltens der Syncromill liefert wichtige Informationen für alle Beteiligten, im Speziellen allerdings für die Softwaretechniker. Durch einen, vor der Programmierung festgelegten Ablauf, können Unklarheiten und damit potentielle Fehler vermieden werden.
  In diesem Schritt werden dem Systemmodell bis auf die Zustände und Ereignisse keine weiteren Informationen hinzugefügt, da auf die bestehenden Funktionen zurückgegriffen wird.

Die Menge an Informationen im Systemmodell ist abhängig von der Phase des Entwicklungsprozesses. Zu Beginn werden die erstellten Objekte neben der Bezeichnung wenig Information enthalten. Mit voranschreitendem Entwicklungsprozess werden die zu Beginn leeren Objekte nach und nach mit Informationen befüllt.
iQUAVIS bietet neben der Erstellung des Systemmodells auch die Möglichkeit, Daten aus dem Systemmodell zu verarbeiten. Am Beispiel der Syncromill wird eine FMEA (Abbildung 4.12) für die in der Wirkstruktur vorhandenen Systemelemente erstellt. Die Darstellung der FMEA Analyse erfolgt in Tabellenform, wobei die Tabelle wiederum nur eine Sicht auf das Systemmodell ist. Demzufolge ändert sich bei jeder Änderung der Wirkstruktur der Inhalt der FMEA. Neben der Systemmodellierung bietet iQUAVIS Funktionen zur Verknüpfung von Systemmodell und Projektmanagement. Am Beispiel der Syncromill werden die Maßnahmen der FMEA als *Aufgabe* definiert. In iQUAVIS werden den definierten Aufgaben Ressourcen zugewiesen und in einem Zeitplan (z.B: Gantt Chart) dargestellt. Damit sind Verantwortlichkeiten für alle Betroffenen im Systemmodell gespeichert und abrufbar. In dieser Arbeit wird der Fokus auf das Erstellen des Systemmodells gelegt. Daher sind die Möglichkeiten, die iQUAVIS zur Verknüpfung mit dem Projektmanagement bietet, nicht detaillierter ausgeführt.

System Element		Functions	Potential Failure	Potential Effect Failure	Potential Cause	RPN				Recommended Action
						Severity	Probability/Likelihood	Detectability	RPN (S*P*D)	
Aufbau W-Achse	Motor W-Achse	Motoren ansteuern	Motor dreht nicht	Achse wird nicht bewegt	Motor defekt	2	1	1	2	Bei Inbetriebnahme Prüfen
		W-Achse positionieren	Motor dreht nicht	Achse wird nicht bewegt	Es wird kein Signal gesendet	2	1	2	4	Bei Inbetriebnahme Prüfen
			Motor dreht nicht	Achse wird nicht bewegt	Fehler im Programmcode	2	2	2	8	Bei Inbetriebnahme Prüfen
					Motor zu schwach dimesioniert	3	2	2	12	Auslegungsberechnung und bei Inbetriebnahme überpürfen
				Bearbeitung wird unterbrochen						
			Motor erreicht die gewünschte Geschwindigkeit nicht	Taktzeit kann nicht eingehalten werden	Motor zu schwach dimensioniert	3	2	2	12	Auslegungsberechnung und bei Inbetriebnahme überpürfen
					Zu bearbeitende Werkstücke sind zu schwer	3	2	1	6	Auslegungsberechnung und überpürfen bei der Inbetriebnahme

Abbildung 4.12: Auszug aus der FMEA der Syncromill

4.6.3 Aufbau des Systemmodells der Motorblockbearbeitung

Im zweiten Systemmodell dieser Arbeit wird ein Teil einer Anlage zur Motorblockbearbeitung modelliert. Im Gegensatz zum Systemmodell der einzelnen Syncromill besteht die Anlage zur Motorblockbearbeitung aus 21 Syncromills, der dazugehörigen Fördertechnik und Prüfeinrichtungen. Für einen Teil der Anlage wird im Folgenden die Anforderungsüberprüfung durchgeführt.
Wie im Systemmodell der Syncromill erfolgt die Modellierung auf Basis der CONSENS Methode. Um die Orientierung im Systemmodell zu erleichtern, wird eine Startseite eingefügt. Die Startseite stellt den *Eingang* in die verschiedenen Bereiche des Systemmodells dar. Um das zu ermöglichen, wird vom Datentyp *Element* eine neue Klasse Names *Startseitenelement* erstellt. Die Startseite ist in Abbildung 4.13 ersichtlich.

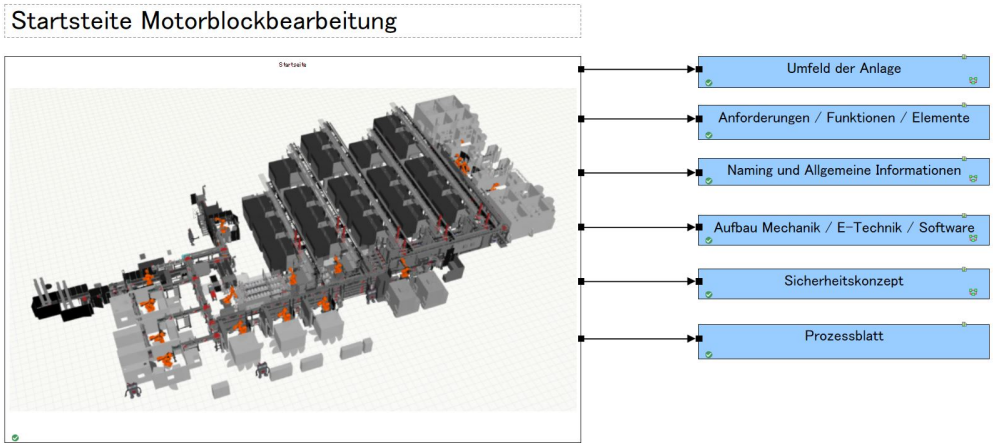


Abbildung 4.13: Startseite als *Eingang* ins Systemmodell

Die in der Abbildung 4.13 dargestellten Elemente führen zu den zugehörigen Detailinformationen. Beispielsweise führt das Element *Umfeld der Anlage* zum Umfeldmodell der Anlage und somit zum ersten Schritt der CONSENS Methode.

Anlage Motorblockbearbeitung - Umfeldmodell

Wie bei der Syncromill werden im ersten Schritt die Umweltweinflüsse der Anlage betrachtet. Darüber hinaus erfolgt bereits im Umfeldmodell eine erste Gliederung der Anlage nach Teilbereichen, um die Navigation durch das Systemmodell zu vereinfachen. Die Art und Weise der Hinterlegung von Informationen erfolgt analog zum Systemmodell der Syncromill. Im Zuge dieser Arbeit wird der Teilbereich der Palettenfördertechnik betrachtet.

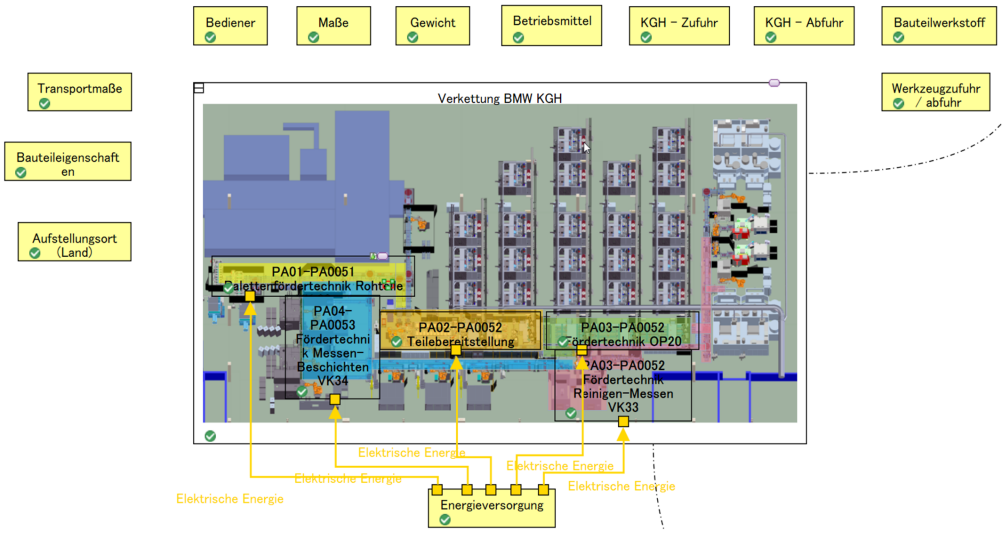


Abbildung 4.14: Umfeldmodell der Anlage

Anlage Motorblockbearbeitung - Anwendungsszenarien

Im zweiten Schritt der CONSENS Methode werden die Anwendungsszenarien festgelegt. In Abbildung 4.15 sind die Anwendungsszenarien abgebildet.
Die Anlage wird dafür ausgelegt, Motorblöcke in den Ausführungen drei, vier und sechs Zylinder zu bearbeiten. Weiters werden die Einlaufphase, das Leerfahren der Anlage und der Servicebetrieb berücksichtigt.

Anlage Motorblockbearbeitung - Anforderungen

Im dritten Schritt der CONSENS Methode werden die Anforderungen ausgearbeitet. Diese werden in der Baumstruktur erstellt und gegliedert. Die Gliederung erfolgt nach den Teilbereichen der Anlage und nach deren Baugruppen. Ebenfalls erfolgt in diesem Systemmodell die Gliederung der Anforderungen mit Hilfe von Tags. Die zu vergebenden Tags legen fest, mit welchem Simulationsprogramm die Anforderung überprüft werden kann. Im Zuge der Anforderungsüberprüfung dieser Arbeit wurden gezielt Anforderun-

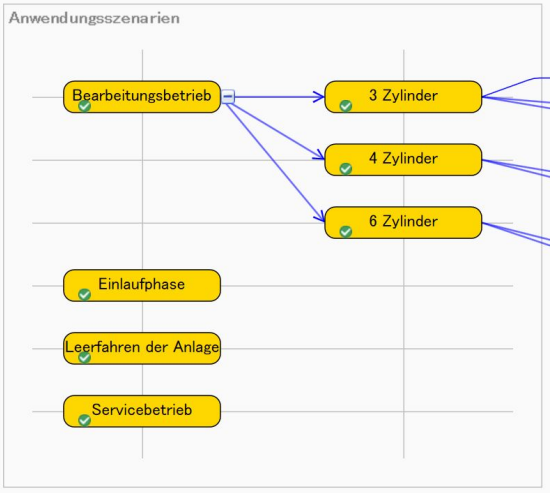


Abbildung 4.15: Anwendungsszenarien der Anlage

gen verwendet, die mit dem Simulationsprogramm *Visual Components* überprüft werden können.
Die in Abbildung 4.16 dargestellten Anforderungen beschreiben Anforderungen an den Teilbereich *Rohteileintransport* der Anlage. Die Anforderungen werden mit den notwendigen Attributen und, wenn möglich, mit Parametern versehen. Diese Parameter werden bei der späteren Anforderungsüberprüfung als Vergleichswert herangezogen. Die in Abbildung 4.16 dargestellten IDs (z.B 5.3.2-1) werden manuell erstellt und dienen ebenfalls der späteren Anforderungsüberprüfung.

Anlage Motorblockbearbeitung - Funktionen
Mit dem vierten Schritt der CONSENS Methode wird die Funktionsstruktur erstellt. Die Erstellung erfolgt analog zur Funktionsstruktur der Syncromill. Die Gliederung dieser erfolgt ebenfalls über die Teilbereiche der Anlage sowie über die Baugruppen.

Anlage Motorblockbearbeitung - Wirkstruktur
Im nächsten Schritt wird die Wirkstruktur modelliert. Die Modellierung erfolgt, wie bei dem Systemmodell der Syncromill, vom Groben ins Detail. Hauptbestandteile der Wirkstruktur sind auch in diesem Fall die Systemelemente, die Funktionslinien und die Ports. Zusätzlich wird zu jedem Elementblockdiagramm eine zugehörige Grafik eingefügt. In einem realen Entwicklungsprozess ist das Einfügen einer Grafik erst möglich, wenn geeignete CAD- Modelle vorliegen. Die Grafik soll dazu dienen, die Systemelemente leichter dem realen System zuordnen zu können. Nachteile treten bei dieser Methode auf, sobald sich die CAD-Modelle ändern. In diesem Fall müsste eine aktualisierte Grafik manuell eingefügt werden. In Abbildung 4.17 ist die Wirkstruktur des Teilbereiches *Rohteileintransport* zu sehen.
Der Nutzen der Grafik wird am Beispiel der Förderer im Teilbereich *Rohteileintransport* deutlich. Auf Grund der zu überbrückenden Länge werden mehrere Förderer nacheinan-

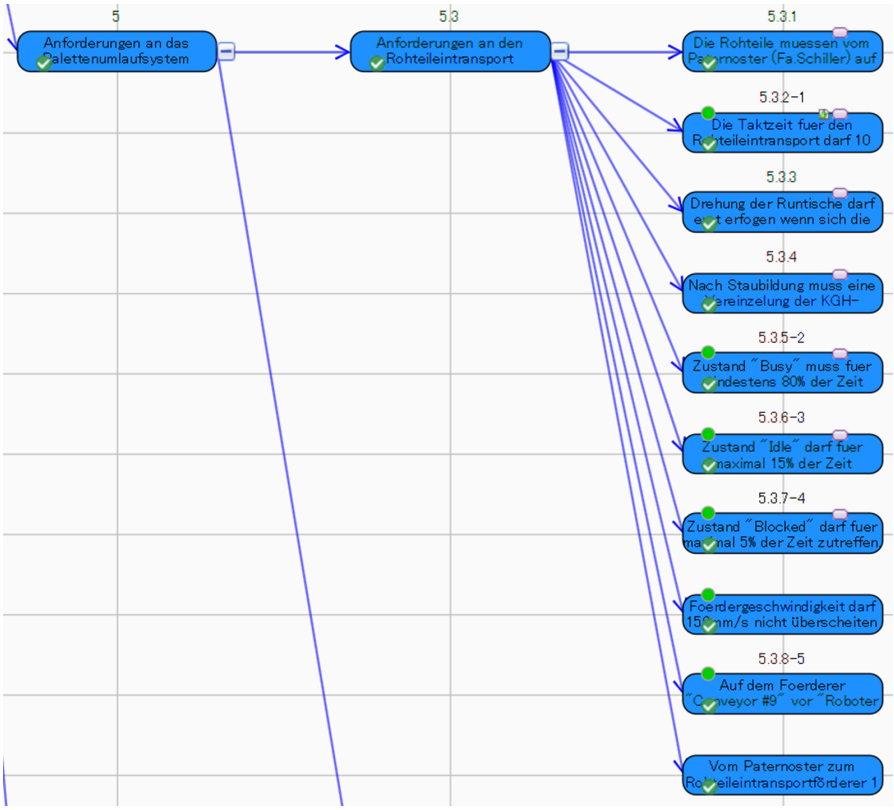


Abbildung 4.16: Ausschnitt der Anforderungen an die Anlage

der eingesetzt. Diese werden üblicherweise durch Nummerierung voneinander unterschieden (z.B: Förderer 1, Förderer 2, Förderer 3, usw.) und dementsprechend so benannt. Ist im Systemmodell keine Grafik vorhanden, ist es beispielsweise im Servicefall schwierig, das richtige Systemelement, in dem die notwendigen Informationen hinterlegt sind, zu identifizieren.

Anlage Motorblockbearbeitung - Verhalten

Im letzten Schritt der CONSENS Methode wird das Verhalten der Anlage modelliert. Hierzu werden die bereits vorhandenen Funktionen in einem Funktionsblockdiagramm dargestellt. Zusätzlich dazu werden Zustandsdiagramme erstellt.
Nach Fertigstellung des Systemmodells werden die vorhandenen Informationen genutzt, um eine FMEA(Abbildung 4.18) zu erstellen. Die Maßnahmen der FMEA werden als Aufgaben ausgeführt und können somit MitarbeiterInnen zugewiesen werden und im Zeitplan hinterlegt werden.

Zwischenfazit

Das Erstellen beider Systemmodelle dient zur Beantwortung der ersten Forschungsfrage (Abschnitt 1.3). Die für den Entwicklungsprozess notwendigen Informationen können im Systemmodell hinterlegt werden, wobei im Bereich der Anforderungen benutzerdefinierte

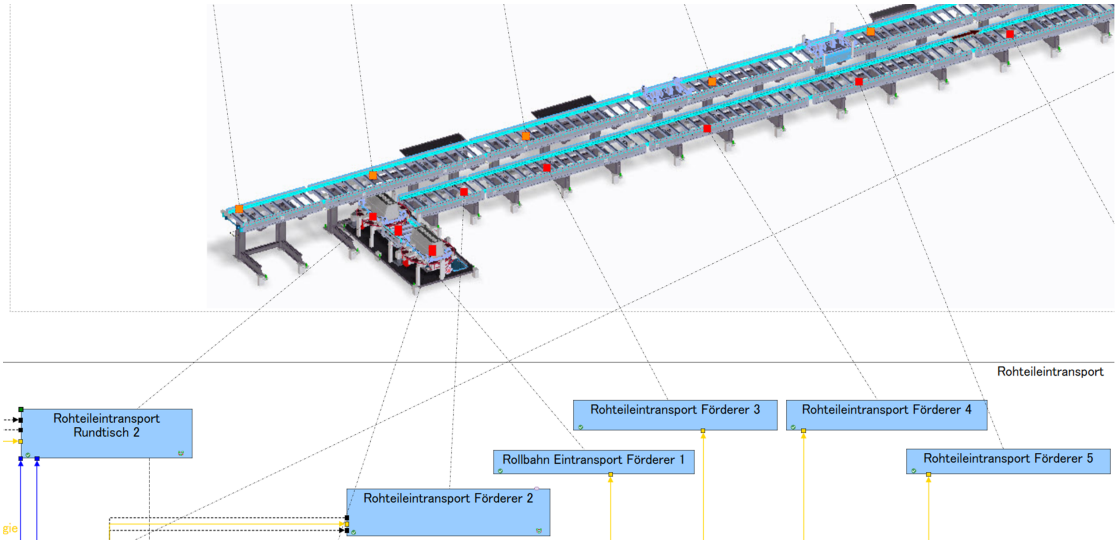


Abbildung 4.17: Wirkstruktur Rohteileintransport (Ausschnitt)

Einstellungen nötig sind, um alle Attribute zu speichern. Da alle Daten über den gesamten Lebenszyklus gespeichert werden können und diese über eine Schnittstelle exportiert und importiert werden können, sind die Systemmodelle als Basis für einen Digitalen Zwilling geeignet.
Zur Beantwortung der zweiten Forschungsfrage (siehe Abschnitt 1.3) wird geprüft, wie die im Systemmodell der Motorblockbearbeitung enthaltenen Anforderungen mit Hilfe eines Simulationsprogrammes automatisiert überprüft werden können.

Systemelement	Funktion			Potentieller Fehler	Potentieller Effekt	Potentielle Ursache	RPN				Vermeidungsmaßnahmen
							Bedeutung	Auftretenswahrscheinlichkeit	Entdeckungswahrscheinlichkeit	RPN (S*P*D)	
Lichtschranke Eintransport Runditsch 1 bei Paternoster Spaltkontrolle ET-023977	Funktionen des Rohteileintransports	Funktionen von Runditsch 1	Spalt kontrollieren Eintransport Runtisch 1 bei Paternoster	Spalt wird als frei erfasst obwohl er nicht frei ist	Bei der Drehung des Rundtisches kann es zu Beschädigungen des Bauteils oder der Anlage kommen.	Sensor defekt	4	1	5	20	Bewährte und zuverlässige Sensoren verwenden. Neue Sensoren vorab testen oder Testberichte der Hersteller anfordern.
						Steuerung falsch programmiert	4	3	2	24	Funktionsablauf schon während der Erstellung mit Softwaretechniker absprechen. Finale Überprüfung bei der Inbetriebnahme
					Stillstand des Rohteileintransports	Sensor defekt	4	1	5	20	Bewährte und zuverlässige Sensoren verwenden. Neue Sensoren vorab testen oder Testberichte der Hersteller anfordern.
						Steuerung falsch programmiert	4	3	2	24	Funktionsablauf schon während der Erstellung mit Softwaretechniker absprechen. Finale Überprüfung bei der Inbetriebnahme
				Spalt wird als nicht frei erfasst obwohl er frei ist.	Rundtisch dreht nicht obwohl er sollte. Dadurch kommt es zum Stillstand des Rohteileintransports.	Sensor defekt	4	1	5	20	Bewährte und zuverlässige Sensoren verwenden. Neue Sensoren vorab testen oder Testberichte der Hersteller anfordern.
						Steuerung falsch programmiert	4	3	2	24	Funktionsablauf schon während der Erstellung mit Softwaretechniker absprechen. Finale Überprüfung bei der Inbetriebnahme

Abbildung 4.18: Auszug aus der FMEA der Anlage

4.7 Aufbereiten der Simulation

Als Simulationsprogramm wird *Visual Components* verwendet. Für die Arbeit liegt bereits eine bestehende Simulation der Anlage vor. Diese Simulation wird dahingehend erweitert, dass eine Anforderungsüberprüfung möglich ist. Vor der Beschreibung der Aufbereitung der Simulation, wird der Aufbau einer Simulation in Visual Components erklärt.

4.7.1 Aufbau einer Simulation in Visual Components

Visual Components ist eine 3D Simulationssoftware zur Fabriksimulation. Eine Simulation setzt sich aus den verschiedenen Komponenten der zu modellierenden Anlage zusammen. Die Visualisierung der Komponenten kann mit vordefinierten Komponenten oder mit CAD-Daten erfolgen. Jeder Komponente können Eigenschaften und Verhalten zugewiesen werden. Weiters sind die Komponenten in der Lage über Schnittstellen zu kommunizieren. Mit der Option, das Verhalten einer Komponente mittels Python Code zu beeinflussen, ist es möglich jede Art von Verhalten abzubilden. In Abbildung 4.19 ist die Anlage im Simulationsprogramm dargestellt.
Für die Visualisierung werden CAD-Daten verwendet. In Abbildung 4.20 wird dargestellt, welche Eigenschaften und Verhaltensweisen die Komponente *Förderer* beinhaltet.
Das am Beispiel des Förderers dargestellte Prinzip wird bei allen Komponenten in der Simulation angewendet. Nach erfolgreicher Simulationserstellung kann das Verhalten der Anlage überprüft werden. Dazu besteht die Möglichkeit, Parameter zu definieren und auszulesen. Im Zuge der Arbeit werden die Parameter definiert, die zur Überprüfung der Anforderungen notwendig sind. Neben den benutzerdefinierten Parametern gibt es noch

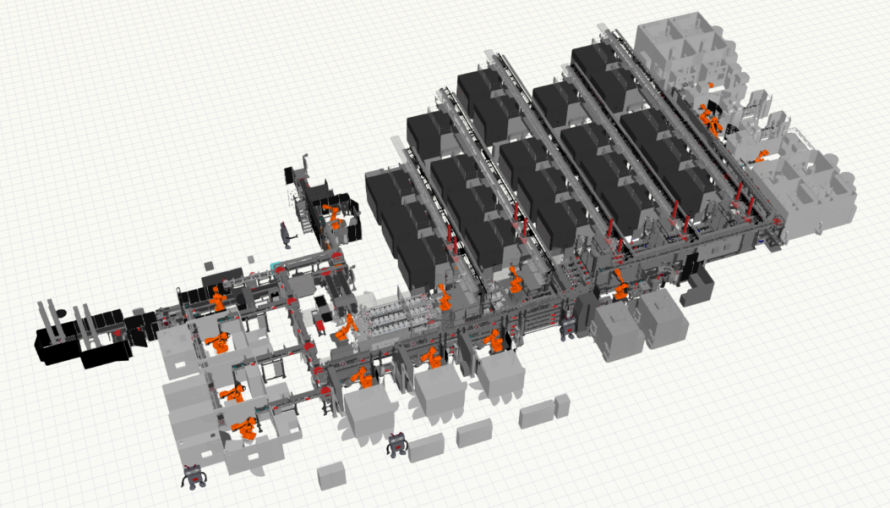


Abbildung 4.19: Aufbau der Anlage im Simulationsprogramm *Visual Components*

die von *Visual Components* vordefinierten Parameter. Diese werden als *Statistikwerte* bezeichnet und beinhalten beispielsweise Informationen über die Auslastung von Komponenten. Sowohl benutzerdefinierte Parameter als auch *Statistikwerte* werden in dieser Arbeit verwendet, um die Anforderungen zu überprüfen.

4.7.2 Ermittlung der Ist-Werte

Im ersten Schritt der Ist-Werte Ermittlung werden die Anforderungen aus dem Systemmodell identifiziert, die mittels Simulation berechnet werden können. Im Zuge dieser Arbeit werden die Anforderungen an den *Rohteileintransport* (Abbildung 4.16) betrachtet. Insgesamt werden im Sytemmodell für den *Rohteileintransport* zehn Anforderungen definiert. Von diesen können sechs mit dem Simulationsprogramm Visual Components überprüft werden. Die Beschreibungen der überprüfbaren Anforderungen lauten:

5.3.2-1 Die Taktzeit der Anlage darf 10 Sekunden nicht überschreiten

5.3.5-2 Zustand „Busy“ muss für mindestens 80 % der Zeit zutreffen

5.3.6-3 Zustand „Idle“ darf für maximal 15 % der Zeit zutreffen

5.3.7-4 Zustand „Blocked“ darf für maximal 5 % der Zeit zutreffen

5.3.8-5 Auf dem Förderer „Conveyor#9“ vor „Roboter Teilebereitstellung“ müssen mindestens 2 Bauteile im Puffer sein

- Fördergeschwindigkeit darf 150 mm/s nicht überschreiten

Jede der genannten Anforderungen weißt einen Parameter auf, der den Soll-Wert darstellt.

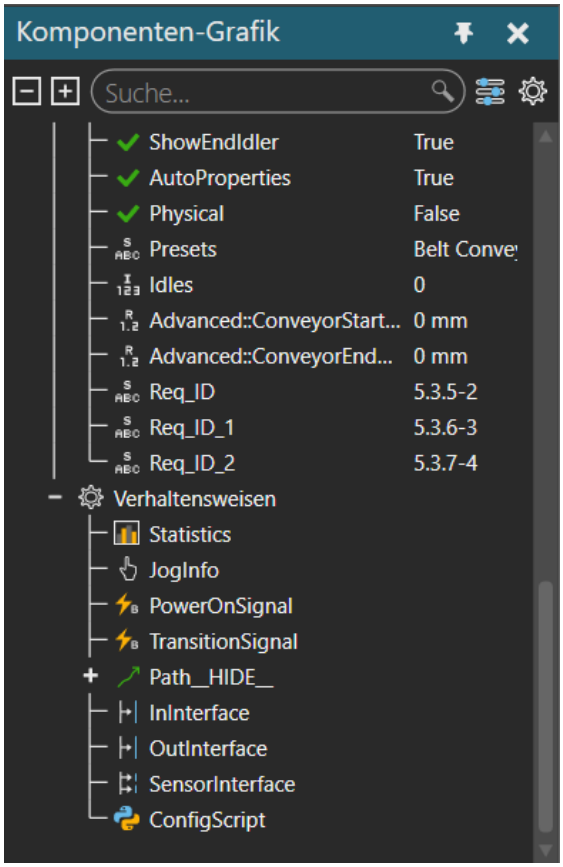


Abbildung 4.20: Eigenschaften und Verhalten der Komponente *Förderer*

Im nächsten Schritt der Ist-Werte Ermittlung werden die Komponenten, bei denen die Anforderungen überprüft werden, mit den dazugehörigen Eigenschaften versehen. In Abbildung 4.20 sind drei hinzugefügte Eigenschaften vom Datentyp String zu sehen. Diese werden mit Req_ID, Req_ID_1 und Req_ID_2 bezeichnet. Die dazugehörigen Werte sind die Anforderungs-IDs aus dem Systemmodell. Diese Strings stellen die Verknüpfung zwischen Visual Components und dem Systemmodell (Abbildung 4.16) dar. Für vordefinierte *Statistikwerte* ist keine weitere Programmierung der Komponenten notwendig. Als Beispiele für *Statistikwerte* können die Zustände *Idle*, *Blocked* und *Busy* genannt werden. Diese können bei jeder Komponente ausgelesen werden.
Die Anforderung *Fördergeschwindigkeit darf 150 mm/s nicht überschreiten* wird im Systemmodell als eine, mit Visual Components überprüfbare, Anforderung getagged. Eine automatisierte Überprüfung ist aber nicht notwendig, da die Geschwindigkeit des Förderers bei dessen Erstellung festgelegt wird und daher keinen errechenbaren Wert darstellt. Zur Ermittlung der Taktzeit der Anlage wird eine Komponente namens *CycleTime Calculator* hinzugefügt. Mit Hilfe dieser wird die Taktzeit ermittelt und als Eigenschaft gespeichert. Nach Durchführung dieser Tätigkeiten sind allen relevanten Komponenten IDs zugewiesen, die eine Anforderung aus dem Systemmodell identifiziert und die Ist-Werte

der Parameter liegen entweder als Statistikwerte oder als benutzerdefinierte Eigenschaft vor.
Im Anschluss daran wird mit der Programmierung der Plugins begonnen.

4.8 Programmierung der Plugins

Damit die Plugins in der angestrebten Art und Weise arbeiten, ist auf beiden Seiten eine Schnittstelle notwendig. Mit *beiden Seiten* ist in dem Fall einerseits das Systemmodell und andererseits das Simulationstool Visual Components gemeint. Seitens des Systemmodells wird die REST API und seitens des Simulationstool die .Net API verwendet. Über die REST API werden die Soll-Werte der Parameter aus dem Systemmodell ausgelesen und über die .Net API des Simulationstools werden die Ist-Werte der Parameter ausgelesen. Damit die betroffenen Komponenten identifiziert werden können, ist die ID notwendig. Der Code zur Identifikation aller Simulationskomponenten, welche eine Req_ID aufweisen, ist in Listing 4.1 dargestellt. Neben der ID muss dem Simualtionstool noch die Information übergeben werden, welcher Parameter ausgelesen werden soll. Dies geschieht über die letzte Zahl der vergebenen ID. In Abbildung 4.20 sind unter anderem drei Identifikationsnummern dargestellt. Die ersten drei Zahlen identifizieren die Anforderung im Systemmodell. Die letzte Ziffer der Identifikationsnummer legt fest, welcher Parameter ausgelesen wird. Am Beispiel der Identifikationsnummer *5.3.5-2* steht der Bereich *5.3.5* für die Anforderung im Systemmodell. Die Zahl 2 am Ende dieser Identifikationsnummer steht für das Auslesen des Zustandswertes *Busy* aus den Statistikwerten. Mit der Festlegung welcher Wert ausgelesen werden soll, wird auch festgelegt, ob für eine erfüllte Anforderung der Ist -Wert größer sein muss als der Soll-Wert oder umgekehrt. Im Anschluss daran kann die Anforderungsüberprüfung durchgeführt werden.

Listing 4.1: Code für die Suche nach Komponenten im Simulationsmodell, welche eine Req_ID aufweisen.

```
foreach (ISimComponent simComponent in _application.World.
    Components)
            {
                list_req_ids.Clear();
                var prop = simComponent.PropertyNamesAndValues;
                foreach (var prop_name_values in prop)
                {
                    if (prop_name_values.Item1.Contains("Req_ID"
                        ))
                    {
                        list_req_ids.Add(prop_name_values.Item2.
                            ToString());
                    }
                }
```

4.9 Anforderungsüberprüfung

In Abbildung 4.21 ist die Vorgehensweise der Anforderungsüberprüfung dargestellt. In dieser Arbeit werden die Anforderungen und deren Soll-Parameter an das Plugin der Anlagensimulation übertragen. Mittels der Anlagensimulation werden die zugehörigen Ist-Parameter ermittelt. Im Anschluss daran werden die beiden Werte verglichen und die Ergebnisse zurück ins Systemmodell übertragen. Durch die Übertragung der Informationen ohne Medienbrüche ist das Systemmodell immer auf dem aktuellsten Stand. Das Systemmodell ist hierbei Ausgangs- und Endpunkt der Anforderungsüberprüfung und bildet somit die Basis für einen Digitalen Informationszwilling.

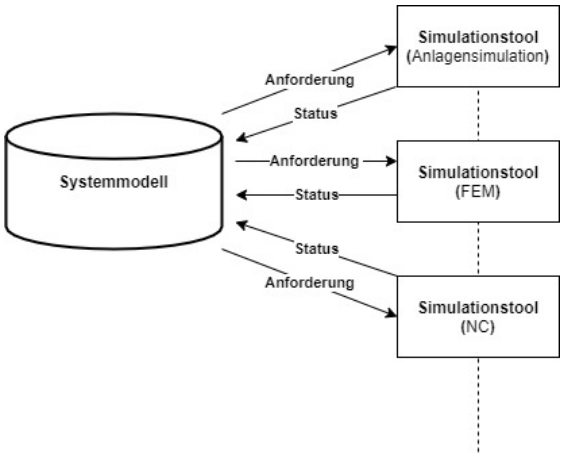


Abbildung 4.21: Vorgehensweise der Anforderungsüberprüfung

In Abbildung 4.22 ist das User Interface mit den importierten Anforderungen aus dem Systemmodell dargestellt. Um zu vermeiden, dass bei einer großen Anzahl an Anforderungen die Übersicht verloren geht, besteht die Möglichkeit, die importierten Daten zu filtern. In Abbildung 4.23 sind die Werte aus der Simulation eingelesen und das Ergebnis der Überprüfung ist dargestellt.

Neben der Anforderungsüberprüfung werden im folgenden Abschnitt noch weitere Möglichkeiten erläutert, wie die Informationen aus dem Systemmodell mit Hilfe von Schnittstellen im Entwicklungsprozess verwendet werden können.

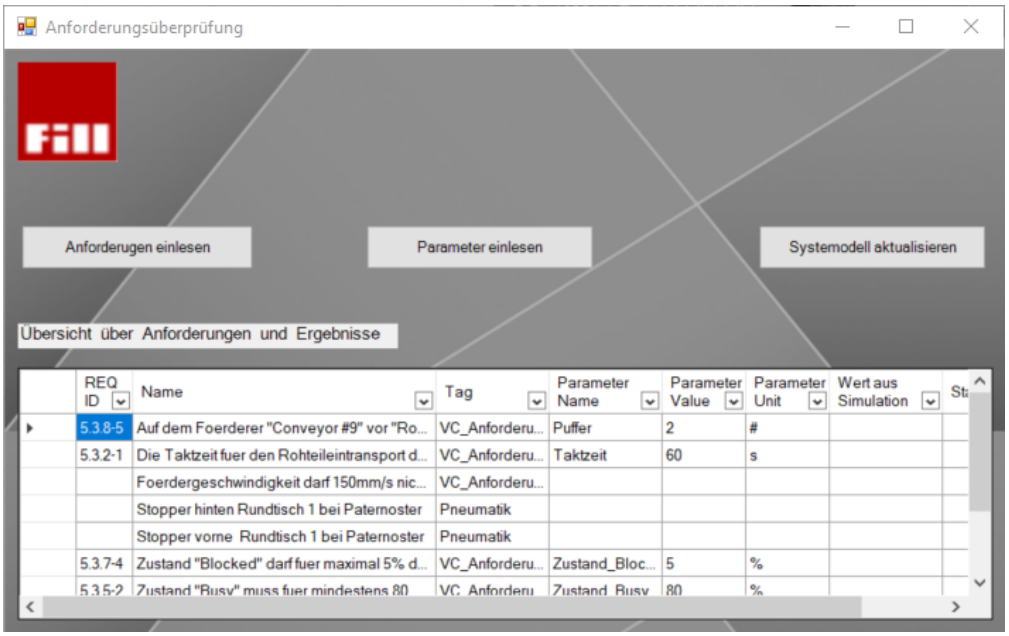


Abbildung 4.22: Importierte Anforderungen aus dem Systemmodell

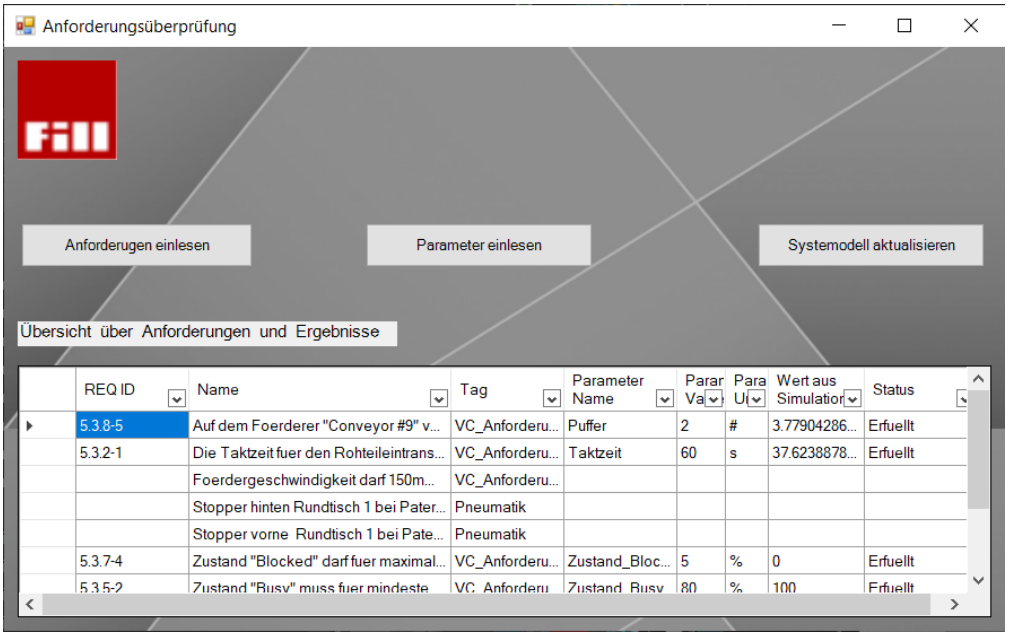


Abbildung 4.23: Parameter eingelesen und Status dargestellt

4.10 Darstellung weiterer Verwendungsmöglichkeiten des Systemmodells

Neben der Anforderungsüberprüfung wird ein weiteres Plugin erstellt, mit dem die Änderungen aus dem Systemmodell für den Ersteller der Simulation mit wenigen Klicks sichtbar werden und Links zu den betreffenden Diagrammen hergestellt werden.
Die Änderungsübersicht soll dazu dienen, Änderungen aus dem Systemmodell darzustellen ohne im Systemmodell danach suchen zu müssen. Die Verlinkung soll dazu dienen, bei Bedarf direkt an die richtige Stelle des Systemmodells zu gelangen. In Abbildung 4.24 ist die Vorgehensweise der Änderungsübersicht dargestellt.

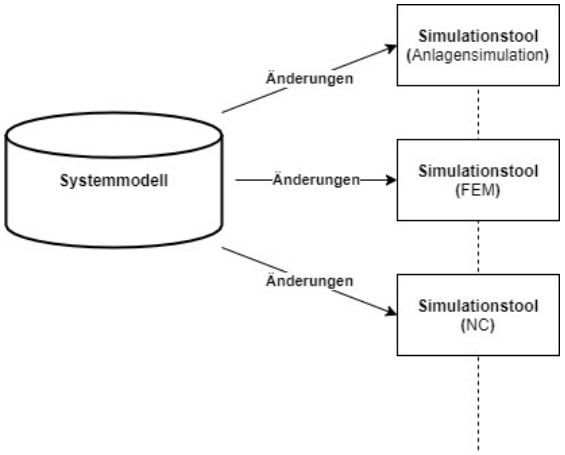


Abbildung 4.24: Vorgehensweise der Änderungsübersicht

In Abbildung 4.25 ist das User Interface der Änderungsübersicht dargestellt. In diesem Fall werden in der Änderungsübersicht hinzugefügte und gelöschte Elemente und Anforderungen dargestellt. Jede angezeigte Änderung ist zur betreffenden Stelle im Systemmodell verlinkt, für den Fall, dass detailliertere Informationen notwendig sind. Somit ist der Ersteller der Simulation in der Lage, ohne Suchaufwand auf diese Veränderungen zu reagieren. Im unteren Bereich werden die Links zu den Funktionsdiagrammen dargestellt. Funktionsdiagramme spielen in der Anlagensimulation ein zentrale Rolle. Es wird entweder auf Basis eines Funktionsdiagrammes der Ablauf der Anlage simuliert, oder es wird mit Hilfe der Simulation der Ablauf festgelegt. In beiden Fällen ist es hilfreich die notwendigen Funktionsdiagramme durch wenige Klicks zu erreichen.

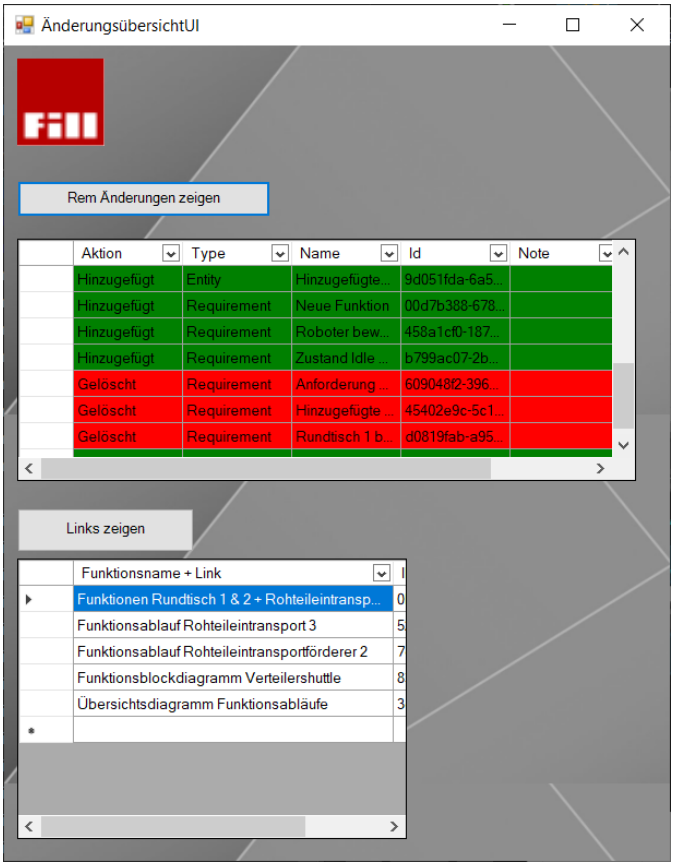


Abbildung 4.25: Änderungsübersicht und Links zu den Funktionsdiagrammen

5 Resultate und Erkenntnisse

Die Beantwortung der ersten Forschungsfrage (Abschnitt 1.3) erfolgt durch das Erstellen von Systemmodellen auf Basis von bestehenden Produkten der Firma Fill. Hierbei treten sowohl bei der Auswahl der notwendingen Faktoren, als auch beim Prozess des Erstellens der Systemmodelle wichtige Erkenntnisse auf.
Als ersten Schritt in Richtung Systemmodell wird die Sprache ausgewählt. Die Anforderungen, die diese erfüllen muss, resultieren aus der Befragung von Mitarbeitern der Firma Fill und der Analyse des Einsatzgebietes. Diese Quellen von Anforderungen führen neben den rein technischen Anforderungen zu Anforderungen auf Grund des Einsatzgebietes (Abschnitt 4.3.1). Zur Bewertung werden die Spezifikationstechnik CONSENS und die Sprache SysML in die engere Auswahl genommen. Im Bereich der technischen Anforderungen erfüllen CONSENS und SysML die Anforderungen. Ein Unterschied ist bei den Anforderungen auf Grund des Einsatzgebietes zu erkennen. Hierbei erfüllt die Spezifikationstechnik CONSENS die Anforderungen besser als die SysML. Die Erkenntnis aus dieser Bewertung ist, dass abhängig vom Einsatzgebiet unterscheidliche Anforderungen an die Sprache gestellt werden und somit auch die Eignung der Sprache variieren kann.
Die Auswahl der verbleibenden beiden Faktoren resultiert in der Spezifikationstechnik CONSENS und dem Werkzeug iQUAVIS.
Im Zuge der Erstellung der Systemmodelle erweist sich die Methode CONSENS als hilfreicher roter Faden, um den Überblick während der Modellierung nicht zu verlieren. Weiters sind die verwendeten Baumstrukturen hilfreich für die Darstellung der Zusammenhänge einer großen Anzahl an Anforderungen, Funktionen und Systemelementen. Durch die Möglichkeit neue Klassen zu erstellen und benutzerdefinierte Ports hinzufügen zu können, steht es dem Ersteller des Systemmodells weitestgehend frei, wie das Systemmodell erstellt wird. Die Klassenstruktur und die vorhandenen Ports haben Auswirkungen auf die Filterbarkeit des Systemmodells und die Weiterverwendung der Daten aus dem Systemmodell. Die Filterbarkeit der Informationen ist im Entwicklungsprozess von zentraler Bedeutung. Für den Fall, dass mehrere Personen an einem Systemmodell arbeiten, braucht es daher Regeln, die das Erstellen benutzerdefinierter Klassen und Ports festlegt. Anforderungen werden hauptsächlich in der Baumstruktur erstellt und bearbeitet. Pflichtattribute, die Anforderungen laut Grande(Abbildung 2.1) aufweisen müssen, können dem Systemmodell hinzugefügt werden, allerdings können die ID und die Version nur behelfsmäßig hinterlegt werden. Dieser Umstand kann bei einer großen Anzahl von Anforderungen zu Unklarheiten und Inkonsistenzen führen. Durch das Hinzufügen von Grafiken zu den Systemelementen, fällt es leichter, ein gemeinsames Systemverständnis zu erreichen. Der Nachteil dieses Vorgehens ist die Notwendigkeit zur manuellen Aktualisierung der Grafiken im Falle einer Änderung bei dem Systemelement.
Als hilfreich für die Orientierung im Systemmodell erweist sich ein definierter *Eingang*

ins Systemmodell (Abbildung 4.13).
Der technische Aufbau des Systems und die notwendigen Daten lassen sich über die vier Datentypen speichern und mit Diagrammen, Tabellen usw. darstellen. Hierbei erweist sich die Möglichkeit der Verlinkung in externe Datenbanken als hilfreich. Verlinkt wird beispielsweise auf das Datenblatt eines Motors oder einer CAD-Datei. Die Links sollen so gestaltet sein, dass immer auf den aktuellsten Stand verlinkt ist. Ein weiterer wichtiger Aspekt ist das Hinzufügen von Parametern und Tags zu den Systemelementen. Beide können benutzerdefiniert erweitert werden und über die Eingabemasken den jeweiligen Systemelementen zugewiesen werden.
Zusammenfassend betrachtet, erfüllen die ausgewählten Faktoren die Anforderungen an die Methode nahezu vollständig. Nicht vollständig erfüllt werden die Anforderungen im Bereich der Anforderungsverwaltung. Weiters werden Anforderungen und Funktionen durch denselben Datentyp, namens Anforderung, dargestellt. Die Funktionen sind somit kein eigener Datentyp, sondern nur ein Objekt vom Datentyp Anforderung. Für die bloße Erstellung des Systemmodells kann dadurch kein nennenswerter Nachteil erkannt werden, allerdings kann es sich bei der Verwendung der Funktionen aus dem Systemmodell im weiteren Entwicklungsprozess, als Vorteil erweisen, wenn Funktionen in einem eigenen Datentyp gespeichert werden können.
Positiv hervorzuheben ist die schnelle Erlernbarkeit des Werkzeuges iQUAVIS. Durch die enge Koppelung von Sprache, Methode und Werkzeug sind die drei Faktoren gut aufeinander abgestimmt.
Im Zuge der Beantwortung der zweiten Forschungsfrage wird ein Plugin in Anlehnung an das Synchronisationskonzept (Abbildung 3.2) programmiert, das die Daten aus dem Systemmodell für die automatische Anforderungsüberprüfung verwendet. Das Plugin ermöglicht die Überprüfung der Parameter der Anforderungen ohne manuelles Zutun. Hier ist anzumerken, dass nur simulierbare Parameter überprüft werden können. Parameter, die mit den vorhandenen Simulationsprogrammen nicht simuliert beziehungsweise berechnet werden können, können auch nicht überprüft werden. Weiters ist anzumerken dass bei erstmaliger Anwendung des Plugins die Verbindung zwischen den Komponenten aus dem Simulationprogramm und den zugehörigen Anforderungen aus dem Systemmodell festgelegt werden muss. Dies kann bei einer großen Anzahl an zu überprüfenden Anforderungen und keiner automatischen ID-Vergabe zu Problemen führen. Über die, für die Arbeit genutzte Schnittstelle, werden die Daten im JSON-Format exportiert. Dadurch ist eine Weiterverwendung der Daten leicht möglich. Hierbei ist anzumerken, dass die Parameter der Systemelemente zwar über die genutzte Schnittstelle exportiert werden können, allerdings nur wenn diese in einer manuell definierten Form vorliegen. Dieser Umstand kann bei einer großen Anzahl an Parametern zu Problemen führen. Wichtig ist zu erwähnen, dass iQUAVIS über weitere Schnittstellen verfügt (e.g. Open Services for Lifecycle Collaboration (OSLC)), welche allerdings in dieser Arbeit nicht verwendet werden.
Im Zuge der Arbeit stellt sich heraus, dass Informationen aus dem Systemmodell bereits vor der Anforderungsüberprüfung verwendet werden können. Das Hauptaugenmerk liegt hierbei auf der Darstellung von Änderungen zum letzten Stand des Systemmodells. Durch die Darstellung von Änderungen im Systemmodell soll erreicht werden, dass alle Projektbeteiligten stets über den neuesten Stand informiert sind.

6 Zusammenfassung und Ausblick

Im Zuge dieser Arbeit werden zwei Systemmodelle erstellt und Plugins programmiert, um die Daten aus dem Systemmodell zu verwenden. Im Vorfeld der Systemmodellerstellung werden die Anforderungen ermittelt, die die Sprache, die Methode und das Werkzeug erfüllen muss. Nach erfolgter Anforderungsermittlung und Auswahl der drei Faktoren (Abschnitt 2.3.3) werden die Systemmodelle erstellt. Das erste Systemmodell stellt ein Bearbeitungszentrum dar und das zweite einen Teil einer verketteten Anlage. Im Zuge der Erstellung wird überprüft, wie sich sich die gewählten Faktoren für die Erstellung mechatronischer Produkte eignen. Die erstellten Systemmodelle erhalten unter Verwendung der CONSENS Methode Informationen über das Umfeld, die Anwendungsszenarien, die Anforderungen, die Funktion, die Wirkstruktur und das Verhalten. Darüber hinaus werden auch *allgemeine* Informationen hinzugefügt, um eine vollständige Abbildung der für den Entwicklungsprozess notwendigen Informationen zu erreichen. Durch den Zugriff auf die Informationen über Schnittstellen, lässt sich das Systemmodell als *Digitaler Informationszwilling* verwenden. Der Zugriff auf die Daten in Form von geeigneten Datenformaten (z.B. JSON) stellt die Grundlage für Anwendungen im Bereich eines Digitalen Zwillings dar.
Im Zuge der Arbeit werden die Daten aus dem Systemmodell für die automatisierte Anforderungsüberprüfung und die Darstellung der Änderungen im Systemmodell genutzt. Hierbei wird der Nutzen eines modellbasierten Ansatzes im Vergleich zu dem dokumentenbasierten Ansatz deutlich sichtbar. Mit dem modellbasierten Ansatz können Informationen ohne Medienbruch über Schnittstellen weitergegeben werden. Weiters ist die Filterbarkeit der Informationen ein erheblicher Vorteil, da die Suchzeiten reduziert werden können.
Das Systemmodell soll nicht als Ansammlung von Diagrammen und Tabellen verstanden werden, sondern als zentrale Ablage und Quelle von Informationen über den gesamten Produktlebenszyklus. In dieser Arbeit werden zur Beantwortung der Forschungsfragen die ersten Phasen des Produktentwicklungsprozesses betrachtet. Im Praxiseinsatz ist es sinnvoll, das Systemmodell über den gesamten Produktlebenszyklus zu verwenden. Hierbei kann sich eine Anbindung an ein PLM-System als hilfreich oder notwendig erweisen. Mit dem modellbasierten Ansatz besteht die Möglichkeit, zeitraubende und fehleranfällige Tätigkeiten, wie das Suchen und Zusammenführen von Informationen zu erleichtern bzw. zu automatisieren. Des Weiteren stellt sich durch die Verwendung von übersichtlichen Darstellungen leichter ein gemeinsames Systemverständnis ein. Dieses spielt gerade in der Anfangsphase von Projekten eine zentrale Rolle. Im weiteren Verlauf des Produktentwicklungsprozesses können Parameter für Berechnungen und die Ergebnisse daraus im Systemmodell zentral verwaltet werden. Es besteht auch die Möglichkeit einer direkten Verknüpfung mit Matlab oder Modelica. Der Nutzen eines Systemmodells soll aber nicht

mit der Fertigstellung des Produkts enden, sondern auch während des Betriebes oder im Servicefall hilfreich sein. Beispielsweise kann im Servicefall das Systemmodell durch den logischen Aufbau helfen, die notwendigen Informationen der betroffenen Komponenten leichter zu finden.
Im Vergleich zum dokumentenbasierten Ansatz gestaltet sich die Wiederverwendung der Informationen aus den Systemmodellen deutlich einfacher. Durch gezieltes Modellieren von Modulen und die Wiederverwendung dieser können Projektdurchlaufzeiten reduziert werden.
Der MBSE Ansatz bietet neue Chancen im Produktentwicklungsprozess. Alleine die Möglichkeit, über Schnittstellen auf Daten aus dem Systemmodell zugreifen zu können, verdeutlicht das Potential dieses Ansatzes. Anders als bei dem Forschungsprojekt mecPro^2 wird in dieser Arbeit kein PLM-System zur Verknüpfung verwendet. Die Verknüpfung von einem PLM-System mit den Informationen aus dem Systemmodell könnte jedoch Gegenstand einer weiteren Arbeit werden. Eine Analyse über die Einsatzmöglichkeiten von Schnittstellen im PEP wäre ebenfalls ein Thema für eine weiterführende Arbeit.
Die vorliegende Arbeit dient der Firma Fill GmbH als Grundlage für weitere Schritte in der Einführung des MBSE Ansatzes. Ziel ist es mit dem modellbasierten Ansatz einerseits in den frühen Phasen der Projekte ein detaillierteres gemeinsames Systemverständnis zu erlangen, andererseits sollen die im Systemmodell gespeicherten Informationen auch für spätere Phasen der Produktentwicklung verwendbar sein.
Für einen Praxiseinsatz ist es ratsam, genau zu definieren, welche Informationen in ein Systemmodell aufgenommen werden sollen. Welche das sind, hängt von den jeweiligen Zielen und den zu modellierenden Produkten ab.

Literatur

[Alt12] O. Alt. *Modellbasierte Systementwicklung mit SysML*. 2012. ISBN: 9783446430662 (siehe S. 16–19, 23–28, 31–35).

[BR16] S. Boschert und R. Rosen. „Digital Twin—The Simulation Aspect“. In: *Mechatronic Futures: Challenges and Solutions for Mechatronic Systems and their Designers*. Hrsg. von P. Hehenberger und D. Bradley. Springer International Publishing, 2016, S. 59–74. ISBN: 978-3-319-32156-1. DOI: 10.1007/978-3-319-32156-1_5 (siehe S. 2, 49).

[EGZ12] M. Eigner, T. Gilz und R. Zafirov. „Proposal for functional product description as part of PLM solution in interdisciplinary product development“. In: *Proceedings of DESIGN 2012, the 12th International Design Conference*. 2012 (siehe S. 43, 44).

[Eig14] M. Eigner. „Einleitung - Modellbasierte virtuelle Roduktentwicklung“. In: *Modellbasierte virtuelle Roduktentwicklung*. Hrsg. von M. Eigner, D. Roubanov und R. Zafirov. 2014. ISBN: 9783662438152 (siehe S. 41–43).

[Eig17] M. Eigner. „Ausgangssituation“. In: *Modellbasierter Entwicklungsprozess cybertronischer Systeme*. Hrsg. von M. Eigner, W. Koch und C. Hrsg. Muggeo. 2017. ISBN: 9783662551240 (siehe S. 1).

[EM17] K. Ehrlenspiel und H. Meerkamm. *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit*. Carl Hanser Verlag GmbH & Company KG, 2017. ISBN: 9783446455450 (siehe S. 15).

[Fil] Fill GmbH. *SYNCROMILL C*. URL: https://www.fill.co.at/de/automotive/metallzerspanungstechnik/standard-bearbeitungsmaschinen/syncromill-c/syncromill-c-direkt-beladen/syncromill-c21-63600/bilder/1511p637i20.html (besucht am 25. 02. 2021) (siehe S. 51).

[FMS08] S. Friedenthal, A. Moore und R. Steiner. *Practical Guide to SysML: Systems Modeling Language*. eng. The MK/OMG Press. San Francisco: Elsevier Science & Technology, 2008. ISBN: 9780123743794 (siehe S. 20, 29–31, 36).

[Gau+08] J. Gausemeier, U. Frank, J. Donoth und S. Kahl. „Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus“. In: *Konstruktion, Ausgabe 7/8-2008* (2008) (siehe S. 36).

[GEK01] J. Gausemeier, P. Ebbesmeyer und F. Kallmeyer. *Produktinnovation: Strategische Planung und Entwicklung der Produkte von morgen*. Hanser, 2001. ISBN: 9783446216310. URL: https://books.google.at/books?id=hiTWwAEACAAJ (siehe S. 36).

[Gra14] M. Grande. *100 Minuten für Anforderungsmanagement*. 2014. ISBN: 9783658064341 (siehe S. 4–14, 85).

[GTS] J. Gausemeier, A. Trächtler und W. Schäfer. „Grundlagen“. In: *Semantische Technologien im Entwurf mechatronischer Systeme*, S. 25–67. DOI: 10.3139/9783446438453.002. eprint: https://www.hanser-elibrary.com/doi/pdf/10.3139/9783446438453.002. URL: https://www.hanser-elibrary.com/doi/abs/10.3139/9783446438453.002 (siehe S. 38).

[Hab+18] R. Haberfellner, O. de Weck, E. Fricke und S. Vössner. *Systems Engineering - Grundlagen und Anwendung*. 14. aktualisierte Auflage 2018. Orell Füssli, 2018. ISBN: 978-3-280-04179-6 (siehe S. 15–17, 61).

[Han+17] M. Hanukaev, C. Huwig, S. Boschert und S. Müller. „System- und Multiphysiksimulation in der CTP Entwicklung“. In: *Modellbasierter Entwicklungsprozess cybertronischer Systeme*. Hrsg. von Martin Eigner, Walter Koch und Christian Hrsg. Muggeo. 2017. ISBN: 9783662551240 (siehe S. 47–49).

[INC] INCOSE. *System and SE Definitions*. URL: https://www.incose.org/about-systems-engineering/system-and-se-definition (besucht am 28. 12. 2020) (siehe S. 15, 17).

[Kai13] L. Kaiser. „Rahmenwerk zur Modellierung einer plausiblen Systemstruktur mechatronischer Systeme“. Dissertation. Fakultät für Maschinenbau, Universität Paderborn, 2013 (siehe S. 22, 36, 37).

[Kal98] F. Kallmeyer. „Eine Methode zur Modellierung prinzipieller Lösungen mechatronischer Systeme“. ISBN 3-931466-41-8. Dissertation. Fakultät für Maschinenbau, Universität Paderborn, 1998 (siehe S. 36).

[Kan+84] N. Kano, N. Seraku, F. Takahashi und S. Tsuji. „Attractive Quality and Must-Be Quality“. In: *Journal of the Japanese Society for quality control* (1984), S. 39–48 (siehe S. 8).

[Kel+09] J. Kelter, S. Reif, W. Bauer und U.E. Haner. *Über die Potenziale von Informations- und Kommunikationstechnologien bei Büro- und Wissensarbeit*. Studie. Fraunhofer Institut, 2009, S. 56 (siehe S. 18, 19).

[MH17] P. Müller und A. Haße. „Gemeinsame Erkenntnislage“. In: *Modellbasierter Entwicklungsprozess cybertronischer Systeme*. Hrsg. von M. Eigner, W. Koch und C. Hrsg. Muggeo. 2017. ISBN: 9783662551240 (siehe S. 46).

[MK17] P. Müller und L. Kirsch. „Vernetzung von Entwicklungsdaten“. In: *Modellbasierter Entwicklungsprozess cybertronischer Systeme*. Hrsg. von M. Eigner, W. Koch und C. Hrsg. Muggeo. 2017. ISBN: 9783662551240 (siehe S. 47, 48).

[PS10] T. Pfeifer und R. Schmitt. *Qualitätsmanagement: Strategien, Methoden, Techniken*. 2010. ISBN: 978-3-446-41277-4 (siehe S. 6).

[Sch18] R. Schlotmann. *Digitalisierung auf mittelständisch*. 2018. ISBN: 9783662557365 (siehe S. 30, 49).

[Two] Twopillars. *Mit CONSENS und iQUAVIS zu effizientem Engineering – Two Pillars*. URL: https://www.two-pillars.de/consens-methode-iquavis/ (besucht am 02. 01. 2021) (siehe S. 37).

[Vaj+17] S. Vajna, C. Weber, K. Zeman, P. Hehenberger, D. Gerhard und S. Wartzack. *CAx für Ingenieure*. 2017. ISBN: 9783662546239 (siehe S. 16).

[Voi13] Jean-Luc Voirin. „Modelling Languages for Functional Analysis Put to the Test of Real Life“. In: *Complex Systems Design & Management*. Hrsg. von Marc Aiguier, Yves Caseau, Daniel Krob und Antoine Rauzy. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, S. 139–150. ISBN: 978-3-642-34404-6 (siehe S. 23).

[Zaf14] R. Zafirov. „Modellbildung und Spezifikation“. In: *Modellbasierte virtuelle Produktentwicklung*. Hrsg. von M. Eigner, D. Roubanov und R. Zafirov. 2014. ISBN: 9783662438152 (siehe S. 21).

[Zin13] Johannes Christian Zingel. „Basisdefinition einer gemeinsamen Sprache der Produktentwicklung im Kontext der Modellbildung technischer Systeme und einer Modellierungstechnik für Zielsystem und Objektsystem technischer Systeme in SysML auf Grundlage des ZHO-Prinzip“. Diss. 2013. DOI: 10.5445/IR/1000037421 (siehe S. 36).

A Inhaltsverzeichnis der CD

Plugin

- **Anforderungsverwaltung -** ***Code für die Anforderungsverwaltung***
- **Änderungsübersicht -** ***Code für die Änderungsübersicht***
- **iQUAVIS-API -** ***Code zum Aufruf der Schnittstelle***
- **app.config**
- **OnInitialize.cs**
- **packages.config**
- **Plugin.csproj**