TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Systems Engineering Funktionen in CAD Systemen

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines

Diplom-Ingenieurs

unter der Leitung von

Univ. -Prof. Dipl.-Ing. Dr. -Ing. Detlef Gerhard

(E307 Institut für Konstruktionswissenschaften und Technische Logistik)

eingereicht an der Technischen Universität Wien

Fakultät für Maschinenwesen und Betriebswissenschaften

Von

Lihui Zhao

Matr.Nr. 0127037

Kennzahl: 066.482

Wien, im Oktober 2013

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Ich habe zur Kenntnis genommen, dass ich zur Drucklegung meiner Arbeit unter der Bezeichnung

Diplomarbeit

nur mit Bewilligung der Prüfungskommission berechtigt bin.

Ich erkläre weiters Eides statt, dass ich meine Diplomarbeit nach den anerkannten Grundsätzen für wissenschaftliche Abhandlungen selbstständig ausgeführt habe und alle verwendeten Hilfsmittel, insbesondere die zugrunde gelegte Literatur, genannt habe.

Weiters erkläre ich, dass ich dieses Diplomarbeitsthema bisher weder im In- noch Ausland (einer Beurteilerin/einen Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Wien, im Oktober. 2013

_____

Unterschrift

Danksagung

Ich bedanke mich besonders bei Herrn Univ. -Prof. Dipl.-Ing. Dr. -Ing. Detlef Gerhard für die Vergabe und Betreuung der Diplomarbeit. Ihre fachliche und organisatorische Unterstützung, sowie ihre Diskussionsbereitschaft und die daraus resultierenden Vorschläge und Anregungen waren eine große Hilfe bei der Erstellung dieser Arbeit.

Ebenfalls möchte ich mich bei Ao. Univ. Prof. Dr. Manfred GRAFINGER bedanken, der für meine Fragen zur Verfügung stand.

Ich bedanke mich weiters bei meinem Freund Lv Xin für die Korrektur der Arbeit.

Nicht zuletzt möchte ich mich bei all jenen bedanken, die mich während des Studiums und bei der Erstellung dieser Arbeit tatkräftig unterstützt haben. Insbesonders meine Eltern, die mich nicht nur finanziell, sondern auch moralisch sehr unterstützt haben und ohne die dieses Studium gar nicht erst möglich gewesen wäre.

Kurzfassung

Die Mechatronik prägt den modernen Maschinenbau. Der Einsatz von Elektronik und Software verursacht die Komplexität der Systementwicklung. Um diese Komplexität zu beherrschen, wird der interdisziplinärer Ansatz „Modellbasiertes Systems Engineering (MBSE)“ vorgeschlagen. Dazu werden zunächst komplette Systeme in der frühen Phase der Produktentwicklung modelliert. Dadurch entsteht ein Systemmodell des zu entwickelnden Systems, welches als Systemspezifikation zur Unterstützung anderer Entwicklungsaktivitäten dient. Hierfür gibt es die spezielle Systemmodellierungssprache wie SysML.

CATIA ist ein CAD-System und wurde ursprünglich zur Darstellung und Simulation der 3D-Modelle verwendet. Die Funktionalität der neuen CATIA Version 6 wurde in Richtung von MBSE Funktionalität erweitert. Es bietet einen RFLP-Ansatz zur Unterstützung der Produktentwicklung, mit dem nicht nur die physikalischen Modelle darstellt und simuliert, sondern auch die Systeme modelliert und simuliert werden können

Im dieser Arbeit wird ein Systemmodell im Sinne von MBSE beispielhaft für ein mechatronisches System „Pedelec“ mit SysML aufgebaut, wobei die Anforderungen, das Verhalten, die Architektur und die Parameter sowie ihre Beziehungen modellhaft beschrieben werden. Und dann wird analysiert, wie dieses Systemmodell mit Funktionen von CATIA Version 6 weiterverarbeitet werden kann. Die weiteren Simulations- und Analysemöglichkeiten in CATIA-System werden untergesucht.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Produkte werden aufgrund der steigenden Markt- und Kundenanforderungen sowie zunehmenden Funktionalität immer technisch komplexer und intelligenter. Gleichzeitig steigen die Kundenerwartungen an Qualität, Zuverlässigkeit und Preis. Die Produktlebenszyklen werden auch immer kürzer. Unter diesen Rahmenbedingungen stehen die Unternehmen vor einer Herausforderung, neue, noch bessere Produkte auf den Markt zu liefern. Bei der Produktinnovation werden eine Verbesserung und oft eine Erweiterung der Funktionalität bestehender Produkte des klassischen Maschinenbaus durch die Integration neuer Lösungsprinzipien weitere Disziplinen durchgesetzt. [1] Klassische mechanische Lösungsprinzipien werden zunehmend hinterfragt und durch mechatronische Lösungsansätze ersetzt.

Mechatronische Produkte entstehen durch die Kombination und Integration von Lösungsprinzipen aus den Fachdisziplinen Mechanik, Elektrotechnik und Informationstechnik. Dies führt dazu, dass die Komplexität der Entwicklungsmethoden, -prozesse sowie der dabei angewandten IT-Werkzeugen und anfallenden Produktdaten zunehmend erhöht wird. [2] Diese Komplexität entsteht dadurch, dass die Technologien der verschiedenen Disziplinen miteinander kombiniert werden müssen. Dafür ist eine spezifische Methode erforderlich, um die komplexen Systeme zu entwickeln. Systems Engineering ist eine interdisziplinäre Methode, mit der das System auf systematische Art und Weise entwickelt werden kann. Dabei wird das aus der Softwareentwicklung bekannte V-Modell (VDI 2206) als Vorgehensweise zur Entwicklung mechatronischer Systeme vorgenommen.

Bei der Systementwicklung hat die frühe Entwicklungsphase eine entscheidende Bedeutung für die folgenden Phasen, da die grundlegende Systemspezifikation hier festgelegt werden muss. Die komplexen Systeme sollen hier vollständig und fehlerfrei spezifiziert werden. Außerdem ist die frühe Entwicklungsphase von besonderer Bedeutung für die entstehenden Kosten während einer Entwicklung und die Dauer der gesamten Entwicklungszeit eines Systems.[3] Nach der „Zehnerregel“ werden die Kosten, die durch Fehler verursacht werden, umso teurer, je später die Fehler im

[1] Vgl. Zirkler, 2010. S.1
[2] Vgl. Bellalouna, 2009, S.3
[3] Vgl. Eckrich, 1996, S.14

Entwicklungsablauf entdeckt werden. Es wird immer versucht, Fehler frühzeitig zu entdecken. Die Entwicklungszeit kann aufgrund ungenügender Planungs- und Entwurfsaktivitäten erhöht werden.[4] Deshalb ist eine Unterstützung in dieser Phase notwendig, um die Komplexität zu beherrschen sowie die Kosten und Zeit zu sparen. Modelle können als Hilfsmitteln in dieser Phase verwendet werden. Durch die Nutzung der Modellbildung und der rechnerunterstützten Analyse werden Kosten- und Zeitvorteile gewonnen.[5]

Die Festlegung, wie eine Produktfunktion realisiert werden soll, erfolgt in der frühen Phase (Systemspezifikation) der Produktentwicklung. Hier sollen die Systeme durch Systemanforderungen, -struktur und -verhalten vollständig beschrieben werden. Nach der Idee der Modellbildung kann ein komplettes System in einem Systemmodell dargestellt werden. Dadurch kann das Produkt frühzeitig funktional beschrieben werden, und die Entwickler können aus verschiedenen Disziplinen ein gesamtes Verständnis vom Produkt generieren. Der Einsatz von Elektronik und Software in mechanische Systeme ermöglicht es, Produktfunktionen nicht nur hardwaretechnisch sondern auch softwaretechnisch abzubilden. Model Based Systems Engineering (MBSE) bietet die Möglichkeit zur modellbasierten Entwicklung der komplexen Systeme, wobei das zu entwickelnde System in einem konsistenten Systemmodell durch die standardisierte Modellierungssprache, z.B. SysML (System Modelling Language) dargestellt werden kann. Dieses Systemmodell umfasst die Anforderungen, Struktur und Verhalten des Systems sowie Parametrik. Es wird mit dem Zweck erstellt, die Beschreibung des Produktes zu erstellen und alle kritischen Aspekte unter Einbeziehung aller Beteiligten zu dokumentieren.

Im Sinne der virtuellen Produktentwicklung sollen die Produkte mit ihren Eigenschaften als digitales Modell im Rechner möglichst vollständig beschrieben werden. Früher hat man nur überwiegend geometrische Modelle mit entsprechender Software erstellt, aber im Rahmen der modellbasierten Entwicklung werden komplette Systemmodelle erstellt. Das heißt, dass der ganze Entwicklungsprozess durchgängig durch Rechner unterstützt werden soll.

Computer-Aided Design (CAD) wurde ursprünglich als ein IT-Werkzeug zu dem technischen Zeichnen verwendet. Heute ist CAD ein komplexes IT-system, mit dem nicht nur der Entwurf und die Konstruktion technischer Lösung, sondern auch die Simulation und Analyse ihrer Verhalten möglich sind, z.B. FEM-Simulationen oder

[4] Vgl. Eckrich, 1996, S.15
[5] Vgl. VDI-Richtlinien 2206, 2004, S.47

kinematische Simulationen. Außerdem können auch die fertigungsrelevanten Abläufe mit CAD simuliert werden. Die Funktionalität vom CAD-System wird immer erweitert, auch in Richtung von MBSE Funktionalität.

Computer Graphics-Aided Three-Dimensional Interactive Application (CATIA) ist ein von Dassault Systèmes entwickeltes CAD/CAM/CAE Package, das zur virtuellen Produktentwicklung dient. Um für Entwicklung interdisziplinärer oder mechatronischer Produkte gerüstet zu sein, bietet CATIA V6 neue Funktionen von der rein physischen Produktdefinition und digitalen Modellierung (digital mock-up) bis hin zur funktionalen Produktdarstellung (functional mock-up) unter Berücksichtigung der verschiedenen Sichten der Produktentwicklung (Anforderung, funktionale, logische und physische Visualisierung). Modellierung der Anforderungen, Funktionen des Systems und Verhalten sowie der Beziehungen zwischen den Komponenten können mit SysML erfolgen. Das dadurch entstehende Systemmodell dient als Grundlage für Weiterentwicklung. Analyse des Systemmodells und des Zusammenhangs zwischen Systemmodell und Funktionen des CATIA V6 ist Gegenstand dieser Arbeit.

1.2 Zielsetzung und Aufgaben der Arbeit

Wegen der Komplexität mechatronischer Systeme sind moderne methodische und rechnerische Unterstützungen in der virtuellen Produktentwicklung erforderlich, insbesondere in der frühen Entwicklungsphase des Systementwurfs.

Die Zielsetzung der Arbeit:

Im Rahmen dieser Arbeit soll ein Systemmodell im Sinne von MBSE mit SysML aufgebaut werden, und analysiert werden, wie es mit Funktionen von CATIA V6 weiter entwickelt werden kann.

Die Aufgaben der Arbeit:

- Erfassen von der Methodik von modellbasierter Systems Engineering (MBSE)
- Analyse der Funktionen von CATIA V6
- Aufbau entsprechender Modelle mit SysML beispielhaft für ein Pedelec, d.h. modellhafte Beschreibung der Komponenten und des Systemverhaltens
- Untersuchung weiterer Simulations-und Analysemöglichkeiten auf der Basis von diesem Systemmodell.

1.3 Aufbau der Arbeit

In Kapitel 2 werden die Grundbegriffe beschrieben. Was ist ein System, und was ist ein mechatronisches System. Dann fokussiert die Arbeit auf die Methoden, welche zur interdisziplinären Systementwicklung dienen. Das Systems Engineering und seine Bausteine Systemstruktur, Systemanforderung und Systemverhalten sowie die systematische Vorgehensweise (V-Modell) für die Entwicklung der mechatronischen Systeme werden vorgestellt. Durch Einsatz vom Systemmodell in SE können die interdisziplinären Systeme modellbasiert entwickelt werden. Das MBSE wird dann als die wichtigste Grundlage dieser Arbeit beschrieben, dazu sein Grundprinzip, das Systemmodell, die Modellierungssprache SysML und das erweiterte V-Modell.

In Kapitel 3 wird ein Überblick über CATIA V6 gegeben. V6 PLM bietet eine kollaborative Systems Engineering (disziplinäre Produktentwicklung) Lösung. Die Key Points von V6 werden beschrieben. Die Funktionen von Modellbildung und Spezifikation sowie die logischen und physikalischen Simulation werden berücksichtigt.

In Kapitel 4 werden die entsprechenden Modelle mit SysML beispielhaft für ein Pedelec aufgebaut. Ein Pedelec -System wird modellhaft beschrieben.

Kapitel 5 beschäftigt sich mit der Untersuchung von weiteren Simulation- und Analysemöglichkeit bei der Anwendung des Systemmodells in CATIA-System.

Schließlich wird in Kapitel 6 die Arbeit mit einer Zusammenfassung abgeschlossen.

2 Grundbegriffe des modellbasierten System Engineerings

2.1 System

Der Begriff des Systems ist in der Literatur aus verschiedener fachlicher Ausrichtung definiert. Nach Steffen ist: „ Ein technisches Produkt ist im Sinne der Systemtheorie ein System“,[6] Die allgemeine Definition vom System nach Patzak lautet:

„Ein System besteht aus einer Menge von Elementen, welche Eigenschaften besitzen und welche durch Relationen miteinander verknüpft sind.“ [7]

Für Systems- und Software Engineering wird System so definiert:

“Ein System ist ein Gebilde (Gefüge, Komplex, Zusammenstellung) von bestimmen Objekten (Komponenten, Bestandteilen, Gegenständen), zwischen denen Beziehungen (Verbindungen, Kopplungen) mit bestimmten Eigenschaften bestehen.“ [8]

Systeme bestehen demnach aus verschiedenen Elementen, die auch Teile, Komponenten, Bestandteile oder Objekte genannt werden können. Diese Elemente besitzen Eigenschaften, die seine Gestalt und Verhalten beschreiben, und damit die Funktionen angeben. Die Beziehungen zwischen den Systemelementen und ihren Eigenschaften verbinden das System zu einem Ganzen, damit eine neue Funktionalität oder Aufgabe erfüllt werden kann, die die einzelnen Elemente nicht erfüllen können.[9]

Systeme werden durch die so genannte Systemgrenze von ihrer Umgebung getrennt. Technische Systeme stehen durch Eingangsgrößen und Ausgangsgrößen mit ihrer Umgebung in Verbindung. Und die Eingangs- und Ausgangsgrößen werden durch Material, Energie, und Information bestimmt. Die Umsetzung von Eingangsgrößen zur den Ausgangsgrößen ist die Funktion eines technischen Systems.[10] So lautet die Definition des technischen Systems in der Konstruktionslehre:

---

[6] Steffen, 2007, S.5
[7] Steffen, 2007, S.5 (zit.nach: Patzak 1982, S.9)
[8] Vogel-Heuser, 2003, S.14 (zit.nach: Bruns 1988)
[9] Vgl. Alt, 2012, S. 8
[10] Vgl. Steffen, 2007, S.5

„ Ein System ist ein abgegrenztes Gebilde, das eine bestimmte Funktion erfüllt.“[11]

Wenn ein Systemelement wieder aus untergeordneten Elementen besteht, kann es als ein selbstständiges System betrachtet werden. Und es wird als Teilsystem eines Systems bezeichnet. Das System hat somit eine hierarchische Struktur.

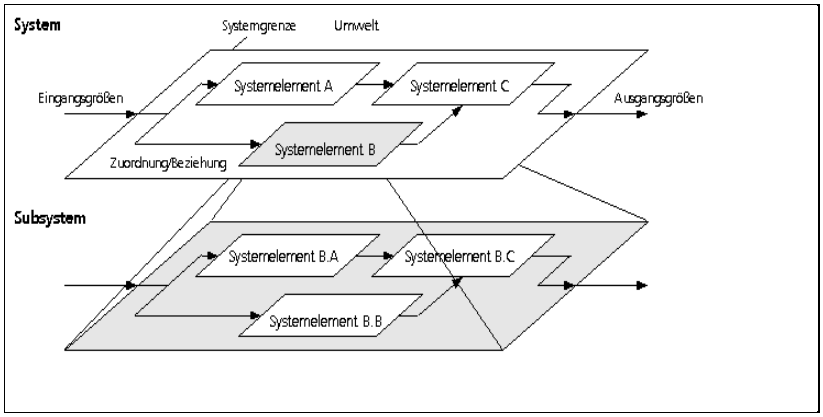

Abbildung 1: Darstellung des technischen Systems [12]

Die Beziehungsvielfalt und die Elementvielfalt beschreiben die Komplexität des Systems. Also steigt die Komplexität eines Systems mit der Anzahl der Elemente, der Anzahl der Beziehungen zwischen diesen Elementen sowie der Art der Beziehungen.[13] Die Kunden erwarten immer mehrere Funktionen von den modernen Produkten. Mehrere Elemente aus verschiedenen Fachdisziplinen werden in einem System integriert. Das bedeutet, dass die Systeme mehrere Elemente und Beziehungen enthalten und damit immer komplexer werden.

2.2 Mechatronisches System

Außer der Anforderung an die Funktionalität der Systeme steigen auch andere Anforderungen stetig für die modernen technischen Produkte, z.B. Qualität, Recycling, Design, Preis usw. Unter diesem Wettbewerbsdruck werden neue Prinziplösungen gefordert, um die Anforderungen zu erfüllen, die Kosten/Nutzen-Verhältnis der Produkte zu verbessern und damit das Erfolgspotential zu gewinnen. Mechatronik ist ein Kunstwort aus Mechanik und Elektronik, das durch die Kombination und Integration von Lösungsprinzipen aus den Fachdisziplinen Mechanik, Elektrotechnik und Informationstechnik entsteht. VDI-Richtlinie 2206 definiert Mechatronik als

[11] Steinschaden, 1998/1999 S.3
[12] http://www.eco2l.de/planung/systematik.html (10.03.2013)
[13] Vgl. http://de.wikipedia.org/wiki/Komplexit%C3%A4t#cite_note-4 (10.03.2013)

„…das synergetische Zusammenwirken der Fachdisziplinen Maschinenbau, Elektrotechnik und Informationstechnik beim Entwurf und der Herstellung industrieller Erzeugnisse sowie bei der Prozessgestaltung“[14]

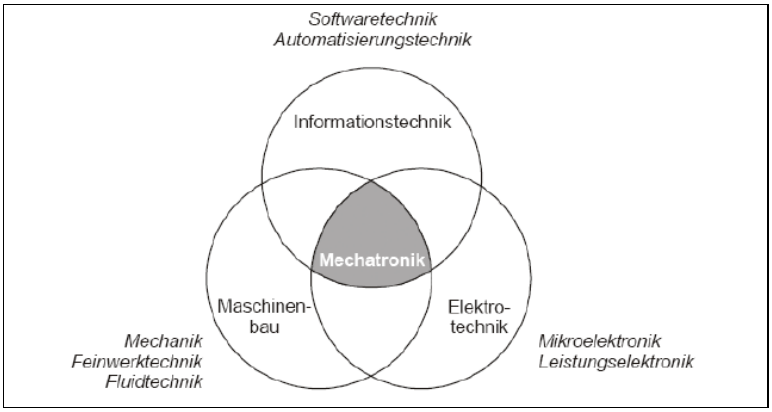

Abbildung 2: Synergie aus dem Zusammenwirken verschiedener Disziplinen [15]

Mechatronische Systeme sind technische Systeme, in denen mechanische, elektronische und informationstechnische Funktionselemente miteinander verknüpft sind. Ein untrennbares Gesamtsystem wird durch diese Verschmelzung anstelle mehrerer Modelle dargestellt.[16] Nach VDI 2206 bestehen mechatronische Systeme aus einem Grundsystem, Sensoren, Aktoren und einer Informationsverarbeitung. Die Grundstruktur eines mechatronischen Systems wird in folgender Abbildung gezeigt.

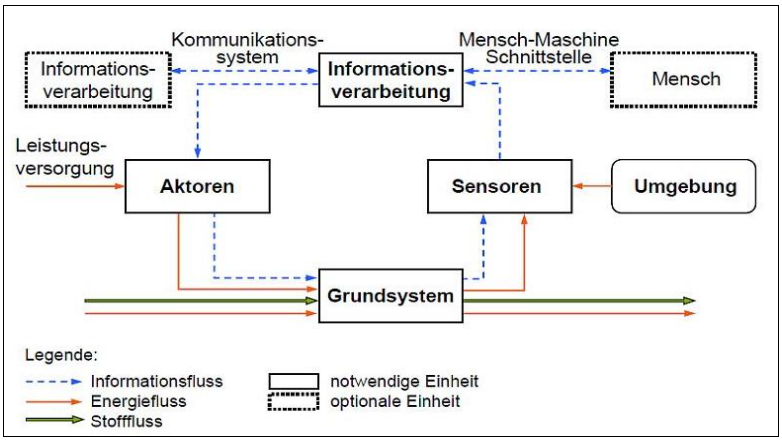

Abbildung 3: Grundstruktur eines mechnatronischen Systems [17]

Das **Grundsystem** kann ein mechanisches, elektromechanisches, hydraulisches oder pneumatisches System bzw. eine Kombination aus diesen sein. In den meisten

[14] Vgl. VDI-Richtlinien 2206, 2004, S.14
[15] Gehrke, 2005, S.10 (zit.nach:Ise99)
[16] Vgl. Dohmen, 2002, S.14
[17] Vgl. VDI-Richtlinien 2206, 2004, S.14

Fällen benutzt man ein physikalisches System als Grundsystem.[18] Es übt die Hauptfunktion des Systems aus.

Sensoren haben die Aufgabe um ausgewählten Zustandsgrößen des Grundsystems zu bestimmen. Sensoren können physisch vorhandene Messwertaufnehmer oder auch reine Softwaresensoren sein. Die Eingangsgrößen werden mittels z.B. Mikroprozessor digitalisiert. Die dadurch erzeugten Signale werden in Informationsverarbeitung übertragen. **Informationsverarbeitung** bestimmt die notwendigen Auswirkungen, um die Zustandsgrößen des Grundsystems in gewünschter Weise zu beeinflussen. Die aus der Informationsverarbeitung gelieferten Signale werden weiter in Aktor übertragen. Diese Signale werden durch **Aktoren** in analoge Größen umgesetzt und verstärkt, und beeinflussen direkt am Grundsystem.[19]

Die Beziehungen zwischen den Komponenten eines mechatronischen Systems sowie zwischen System und seiner Umgebung lassen sich durch drei Flüsse darstellen:[20]

- **Stoffflüsse**: Stoffe, die Komponenten eines mechatronischen Systems miteinander verbinden, sind z.B. feste Körper, Prüfgegenstände, Behandlungsobjekte, Gase oder Flüssigkeiten.
- **Energieflüsse**: Unter Energie ist in diesem Zusammenhang jede Energieform zu verstehen. Wie z.B. mechanische, thermische oder elektrische Energie, aber auch Größen wie Kraft oder Strom.
- **Informationsflüsse**: Informationen, die zwischen den Komponenten mechatronischer Systeme ausgetauscht werden, sind beispielsweise Messgrößen, Steuerimpulse oder Daten.

Komplexe mechatronische Systeme bestehen im Allgemeinen aus der synergetischen Integration verschiedener mechatronischer Module. Jedes Modul ergibt sich aus einer Gruppe von zusammengefassten Elementen, und erfüllt eine bestimmte Funktion.[21] Die oben beschriebene Grundstruktur kann als mechanisches Funktionsmodul bezeichnet werden. Deshalb setzen sich mechatronische Systeme nicht nur aus einer Grundstruktur zusammen, sondern aus mehreren. Um diese Komplexität der mechatronischen Systeme bewältigen zu können, wurde eine hierarchische Struktur von Lückel vorgeschlagen. (siehe Abbildung 4)

[18] Vgl. VDI-Richtlinien 2206, 2004, S.14
[19] Vgl. VDI-Richtlinien 2206, 2004, S.15
[20] Bellalouna, 2009, S.16 (zit.nach: Gausemeier, Lückel, 2000)
[21] Vgl. VDI-Richtlinien 2206, 2004, S.16

Die Grundstruktur eines mechatronischen Systems versteht man als mechatronisches Funktionsmodul (MFM), das in der untersten Ebene steht. Informationstechnische und/oder mechanische Verkoppelung mehrerer mechatronischen Funktionsmodulen stellt die nächste, höhere hierarchische Ebene dar. Diese Ebene wird als autonome mechatronische Systeme (AMS) bezeichnet. Auf dieser Ebene mit einer zusätzlichen Informationsverbreitung werden übergeordnete Aufgaben wie Überwachung mit Fehlerdiagnose und Instandhaltungsentscheidungen realisiert sowie Vorgaben für untergeordnete mechanische Funktionsmodule generiert. Die oberste Ebene bildet die vernetzten mechatronisches Systeme (VMS), in denen die autonomen mechanischen Systeme über Informationsverarbeitung verkoppelt werden.[22] z.B. Feder- und Neigemodul ist ein MFM. Der einzelne Shuttle „Railcab“ wird als AMS betrachtet, das durch informationstechnische und mechantronische gekoppelte MFM entsteht. Und der Konvoi wird als ein VMS bezeichnet, das durch rein informationstechnische gekoppelte AMS entsteht.

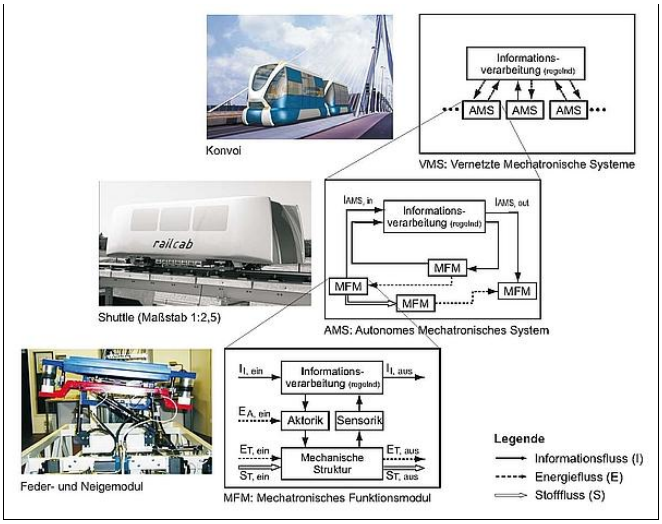

Abbildung 4: Strukturierung mechatronischer Systeme [23]

Um die komplexen Systeme erfolgreich zu entwickeln, muss eine spezifische Methode bereitgestellt werden. Diese Methode entspricht Systems Engineering.

2.3 Systems Engineering

Um die Komplexität des Systems zu beherrschen, braucht man eine methodische Unterstützung bei der Systementwicklung. Systems Engineering (SE) ist eine interdisziplinäre Methode, mit der ein komplexes System auf systematische Art und

[22] Vgl. Steffen, 2007, S.38
[23] http://www.transmechatronic.de/start/mechatronik/?tx_felogin_pi1%5Bforgot%5D=1 (27.03.2013)

Weise entwickelt werden kann. Die Definition des Systems Engineerings nach INCOSE (International Council on Systems Engineering) lautet:

„Das Systems Engineering konzentriert sich auf die Definition und Dokumentation der Systemanforderungen in der frühen Entwicklungsphase, die Erarbeitung eines Systemdesigns und die Überprüfung des Systems auf Einhaltung der gestellten Anforderungen unter Berücksichtigung des Gesamtproblems: Betrieb, Zeit, Test, Erstellung, Kosten & Planung, Training & Support und Entsorgung“ [24]

Systems Engineering beschreibt einen einheitlichen, strukturierten Prozess zur Entwicklung eines komplexen technischen Systems, von Konzeption über die Produktion bis hin zum Betrieb und manchen Fällen bis zur Abbau und Wiederverwertung. In diesem Prozess werden alle Ingenieursdisziplinen integriert. Bei der Entwicklung werden sowohl technische als auch wirtschaftliche Aspekte berücksichtigt.[25]

Systems Engineering umfasst im Wesentlichen das Management und die technische Tätigkeiten, die notwendig sind, um ein komplexes System erfolgreich zu entwickeln. Dazu gehören die Aufgaben: Projektmanagement, Anforderungsanalyse, Anforderungsdefinition, Anforderungsmanagement, Systemdesign, System -verifikation, Systemvalidierung, Systemintegration, Konfigurationsmanagement und Risikomanagement.[26] Außerdem ist ein Prozessmodell auch wichtig, mit dem die Systeme systematisch entwickelt werden können.

Im Systems Engineering existiert eine Vielzahl von Prozessmodellen, die im Laufe der Jahre entwickelt wurden. Das einfache Prozessmodell besteht aus drei Schritten, die nicht nur linear, sondern auch mit Rückkopplungen durchlaufen werden. Im ersten Schritt werden die Systemanforderungen spezifiziert und die zu Stakeholder-Bedürfnissen entsprechenden Komponentenanforderungen bearbeitet und verteilt. Dann werden die Systemkomponenten entworfen, implementiert und getestet, um sicherzustellen, dass sie die Systemanforderungen erfüllen können. Im letzten Schritt werden die Systemkomponenten zu einem Ganzen integriert, und wird das System übergeprüft, ob die Systemanforderungen erfüllt sind.[27] Abbildung 5 stellt den einfachen Entwicklungsprozess dar. Und für mechatronische Systementwicklung

[24] Weilkiens, 2008, S.11 (zit.nach: INCOSE, 2004)
[25] Vgl. Weilkiens, 2008, S.11f
[26] Vgl. Weilkiens, S.13
[27] Vgl. Friedenthal; More; Steiner, 2012. S.4

ist das so genannte V-Modell besonders geeignet, es wird in Kapitel 3.2.1 detailliert vollgestellt.

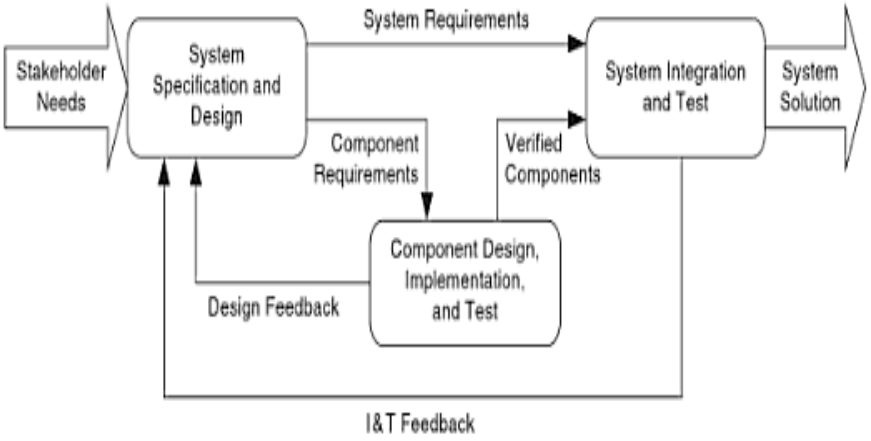

Abbildung 5: Einfaches Systems Engineering Prozessmodell [28]

Die Hauptaufgabe in der frühen Systementwicklungsphase ist das zu entwickelnde System vollständig zu beschreiben. Eine Systemspezifikation wird zunächst erstellt. Auf der Basis der Systemspezifikation wird das System weiter konkretisiert bzw. entwickelt. Diese Systemspezifikation umfasst die Systemanforderung, die Systemstruktur, und das gemäß der Systemanforderung erstellte Systemverhalten. Aus der Zusammensetzung von Systemanforderungen, Systemarchitektur und Systemverhalten ergibt sich ein Bild von dem zu entwickelnde System. Systemanforderungen, Systemarchitektur und Systemverhalten werden als die Bausteine des Systems Engineering betrachtet.[29]

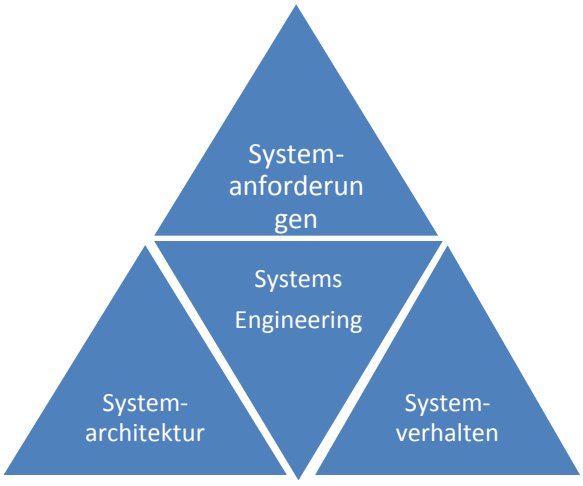

Abbildung 6: Die drei Bausteine des SE

[28] Friedenthal; More; Steiner, 2012. S.4
[29] Vgl. Alt, 2012, S.9

2.3.1 Systemanforderungen

Systemanforderungen sind die Texte, die die Funktionalitäten, die Eigenschaften und die Beschränkungen des zu entwickelnden Systems definieren.[30] Sie beantworten die beiden Fragen „ Was soll das System tun“ und „ Wie soll sich das System in Bezug auf die wahrnehmbaren Eigenschaften wie Zuverlässigkeit, Benutzbarkeit, Effizienz, Wiederverwendbarkeit, und Übertragbarkeit usw. verhalten“. Nach diesen zwei Fragen kann man die Anforderungen in zwei Klassen untergliedern, die funktionalen Anforderungen und die nichtfunktionalen Anforderungen.[31]

Die funktionalen Anforderungen spezifizieren das Verhalten des Systems, bzw. der Systemkomponenten, d.h. die Funktionalität, die von einem System erwartet wird. Ein System könnte durch das Zusammensetzen von funktionalen Anforderungen und Architektur vollständig beschrieben werden.[32]

Die nichtfunktionalen Anforderungen sind die Anforderungen, welche die Umstände beschreiben, unter denen die geforderte Funktionalität zu erbringen ist.[33] Sie sind nur indirekt für die Funktion des Systems entscheidend und können weiter in Leistungsanforderungen, Qualitätsanforderungen und Randbedingungen gegliedert werden. Leistungsanforderungen geben einen gewünschten Bereich für z.B. Zeit, Raum oder Geschwindigkeiten an. Qualitätsanforderungen beschreiben die Qualität eines Systems, z.B. Zuverlässigkeit, Benutzbarkeit oder Änderbarkeit usw. Randbedingungen definieren einen beschränkten Lösungsraum. Die Lösungen aus diesem Lösungsraum müssen die geforderte Funktionalität erfüllen.[34] Nicht nur die funktionale Anforderungen sondern auch die nichtfunktionale Anforderungen beeinflussen die Architektur.[35]

[30] Vgl. Käster; Ostermann, Folien 10
[31] Vgl. Liang, 2012, S.11
[32] Vgl. Alt, 2012, S.11
[33] Vgl. https://files.ifi.uzh.ch/rerg/arvo/ftp/ses/kapitel_12.pdf (28.03.2013)
[34] Vgl. Liang, 2012, S.12
[35] Vgl. Alt, 2012, S.11

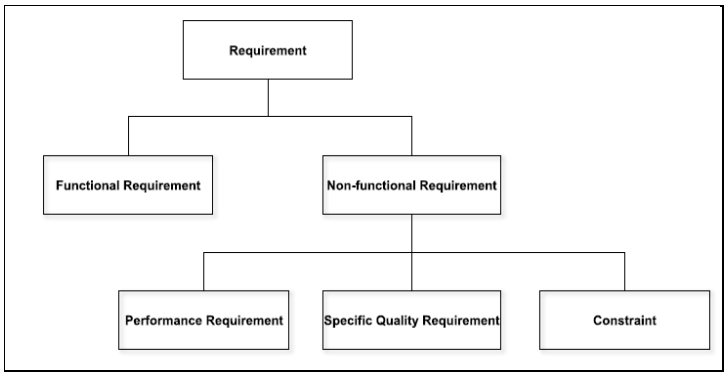

Abbildung 7: Klassifikation der Anforderungen [36]

Bei der Entwicklung muss berücksichtigt werden, dass Kundenanforderungen, Marktanforderungen oder auch gesetzliche Anforderungen als externe Anforderungen betrachtet werden. Die externen Anforderungen können nicht direkt als Entwicklungsgrundlage dienen, da die externen Anforderungen das zu entwickelnde System auf abstraktem Niveau beschreibt. Daher sollen externe Anforderungen immer in interne Anforderungen umgewandelt werden (siehe Abbildung 8). Typischerweise enthalten Lastenhefte externe Anforderungen und Pflichtenhefte interne Anforderungen. Das Pflichtenheft ist oft als die rechtsverbindliche Entwicklungsgrundlage zwischen den einzelnen Entwicklungsteams zu übertragen, da es die technische Lösung definiert und spezifiziert.[37]

2.3.2 Systemarchitektur

Der zweite Baustein Systemarchitektur beschreibt die Struktur des Systems. Dabei werden die Komponenten, aus denen das System bestehen, festgelegt. Die Beziehungen, die zwischen den Systemkomponenten stehen, sowie die Wechselbeziehung zur Umwelt werden definiert. Darüber hinaus wird es festgelegt, wie die Material-, Energie-, und Informationsfluss das System durchlaufen kann.[38] Systemarchitektur zeigt, wie die Systemelemente miteinander verknüpft sind und zusammenarbeiten.

Eine Architektur bildet nur eine mögliche Lösung für ein gegebenes Problem. Für dieses Problem können vielleicht mehre Lösungen existieren, die unterschiedliche Architekturen haben. Mit Hilfe der Architektur können Funktionen aber nicht beschreiben werden, weil die Architektur nur eine statische Sicht des Systems liefert.

[36] Liang, 2012, S.12
[37] Vgl. Alt, 2012, S16f
[38] Vgl. Alt, 2012, S9

Das Verhalten des Systems braucht man, um eine Funktion zu erfüllen. Folgende Fragen könnten mit Architektur beantwortet werden:[39]

- Auf welcher Art und Weise wird eine geforderte Funktionalität realisiert?
- Aus welchen Teilen besteht das zu realisierende System?
- Ist das System mit der gewählten Architektur in der Lage, eine geforderte (neue) Funktionalität zu erfüllen?

2.3.3 Systemverhalten

Das Systemverhalten kann man als eine Systemfunktion verstehen, die durch die Transformation der Eingangsgrößen zu den Ausgangsgrößen besteht.

Die funktionalen Anforderungen beschreiben das Verhalten eines Systems als Text, also informelle Beschreibung. Die Anforderungen müssen von dem Ingenieur gelesen und entsprechend in eine Realisierung des Systems umgesetzt werden. Dafür braucht man eine formale Verhaltensbeschreibung, mit der die Erstellung der Arbeitsprodukte und der Realisierung automatisiert werden kann. Das heißt, dass zeitliche bzw. logische Abläufe der Systemfunktionen, wie sie in den Anforderungen formuliert wurden, festgelegt werden. Verschiedene Betriebszustände des Systems werden hier berücksichtigt.

Diese Aufgaben können rechnergestützt durchgeführt werden, wodurch die Arbeit der Entwicklungsingenieure entlastet wird. Um Systemverhalten formal zu beschreiben, sind viele Beschreibungssprachen entwickelt worden, z.B. SysML, Petri-Netze, oder Grafische Funktionsentwicklung, die aus verschiedenen spezifischen Anwendungsbereichen entstammen.[40] Im Rahmen der Arbeit wird die SysML, die speziell für Systementwicklung ist, verwendet.

2.3.4 Zusammenhänge der Bausteine

Im Systems Engineering sind die drei Bausteine notwendig und voneinander abhängig. Anforderungen und Architektur gehören immer zusammen, weil es eine Wechselwirkung zwischen Anforderungen und Architektur gibt. Die Entscheidungen über Anforderungen beeinflussen die Architektur, und die Entwurfsentscheidungen

[39] Alt, 2012, S.10
[40] Vgl. Alt, 2012, S.14f

beeinflussen wiederum die Anforderungen.[41] Die Architektur muss eine Lösung darstellen, um die von den Anforderungen beschriebenen Funktionen zu erfüllen. Aufgrund dieser Lösung können neue Anforderungen (z.B. die Anforderungen an die in der Architektur definierten Unterkomponenten) entstehen.[42]

Systemverhalten und Systemanforderungen können auch nicht getrennt werden, weil das Verhalten nur auf der Basis der informellen Beschreibung formal beschrieben werden kann. Es gibt somit auch eine Wechselwirkung zwischen Verhalten und Architektur ebenso wie zwischen Anforderungen und Architektur.

Um die Systemkomplexität zu beherrschen, werden Systeme in mehrere Abstraktionsstufen strukturiert.[43] Anforderungen, Architektur und Verhalten können entlang verschiedenen Abstraktionsstufen verfeinert werden, wodurch sie von einer abstrakten Spezifikation bis hin zu einer technologiespezifischen Spezifikation konkretisiert werden. Systems Engineering ist damit ein fortlaufender Prozess. Die drei Bausteine sind miteinander verkoppelt, und entwickeln sich iterativ weiter.[44] Die SE-Bausteine und ihre Zusammenhänge bildet das Systems- Engineering-Schema ab:

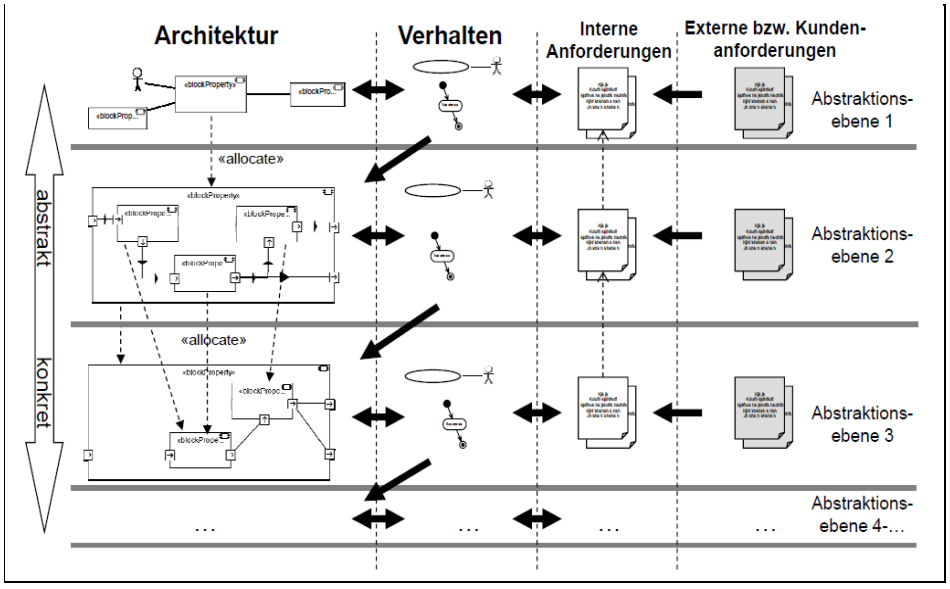

Abbildung 8: Systems-Engineering-Schema [45]

Das Systems-Engineering-Schema zeigt den allgemeinen Lösungsweg des SE, wie ein Lösungskonzept vom Abstrakten bis zum Konkreten entwickelt wird. Dieses

[41] Vgl. Sikora, 2011, S.4
[42] Vgl. Alt, 2012 S. 12
[43] Vgl. Sikora, 2011, S.4 (zit.nach: Pohl, 2008)
[44] Vgl. Alt, 2012 S.16
[45] Alt, 2012 S.16

Schema stellt nur ein Teil von der Systementwicklung dar. Danach folgen noch die qualitätssichernden Maßnahmen wie Reviews, statische und dynamische Tests sowie Prüfungen und Erprobungen, mit denen die Funktionalität des Systems übergeprüft wird. Ein gesamter Entwicklungsprozess ist daher erforderlich, um die Systeme in der logischen Abfolge systematisch zu entwickeln.

Abbildung 5 zeigt nur den einfachen, allgemeinen Entwicklungsprozess. Für bestimmte Systeme braucht man noch einen entsprechenden Entwicklungsprozess, der an den Anforderungen und Eigenschaften dieser Systeme angepasst werden müssen, damit sie unter den Rahmbedingungen Zeit, Kosten und Qualität vollständig und fehlerfrei entwickelt werden können.

2.4 Interdisziplinärer Entwicklungsprozess

Die Komplexität der mechatronischen Systeme entsteht aus einer großen Anzahl von verkoppelten Elementen, die in unterschiedlichen Disziplinen realisiert werden.[46] Bislang sind die meisten Entwicklungsprozesse disziplinspezifisch, d.h. sie werden von den einzelnen Disziplinen dominiert. Nur die Anforderungen und die Eigenschaften einer Domäne werden berücksichtigt. Die Entwicklung der mechatronischen Systeme wird demzufolge getrennt in den involvierten Domänen durchgeführt. Dadurch können langwierige Iterationen, Zeit-, Kosten- und Qualitätsproblem verursacht werden. Dafür ist ein disziplinübergreifender Entwicklungsprozess notwendig, der die Aufgaben, Probleme und Anforderungen aller Disziplinen berücksichtigt.[47] Dieser Entwicklungsprozess muss die verschiedenen disziplinspezifischen Entwicklungsmethoden systematisch und synergetisch zusammenführen, damit die disziplinübergreifende Kommunikation und Kooperation zwischen den beteiligten Fachdisziplinen ermöglicht wird.

2.4.1 V-Modell

Die VDI-Richtlinie 2206 beschäftigt sich mit der systematischen Entwicklung mechatronischen Systeme. Ein Vorgehensmodell wird im Rahmen dieser Richtlinie vorgeschlagen, wobei eine Vorgehensweise, die auf der Grundlage eines allgemeinen Problemlösungszyklus aus dem Systems Engineering basiert, als Mikrozyklus zum Bearbeiten der einzelnen Teilaufgaben dient, und ein V-Modell, das

[46] Vgl. VDI-Richtlinien 2206, 2004, S.4
[47] Vgl. Gehrke, 2005, S.12

im Software Engineering bekannt ist, als Makrozyklus zum Entwickeln des gesamten Systems dient. Hier konzentrieren wir uns auf das Makrozyklus V-Modell.

Das V-Modell stammt aus dem Bereich der Softwareentwicklung und wurde an mechatronische Systementwicklung angepasst.[48] Das V-Modell integriert die an der Entwicklung mechatronischer Systeme beteiligten disziplinspezifischen Prozesse (Mechanik, Elektronik und Informationstechnik), und beschreibt ein generisches Vorgehen beim Entwurf mechatronischer Systeme (Abbildung 9).[49] Dieses Modell umfasst folgende Entwicklungsphasen:[50]

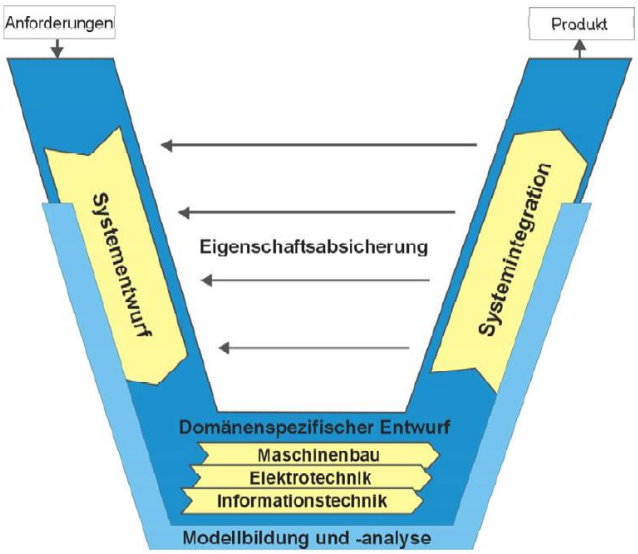

Abbildung 9: V-Modell als Makrozylus [51]

Anforderungen

Ein konkreter Entwicklungsauftrag stellt den Ausgangspunkt dar. Die Aufgabestellung, die definiert, was das System leisten bzw. welche Anforderungen das System erfüllen soll, wird in Form eines Pflichtenhefts beschrieben. Das Produktpflichtenheft wird als Ergebnis dieser Phase erstellt. Dieses Pflichtenheft dient auch als Maßstab zur Bewertung des späteren Produkts.

Systementwurf

Ein domänenübergreifendes Lösungskonzept wird in dieser Phase festgelegt, das die wesentlichen, physikalischen und logischen Wirkungsweisen des zukünftigen

[48] Vgl. VDI-Richtlinien 2206, 2004, S.26
[49] Vgl. Bellalouna, 2009, S.20
[50] Vgl. VDI-Richtlinien 2206, 2004, S.26ff
[51] Vgl. VDI-Richtlinien 2206, 2004, S.29

Produktes und die Art und Anordnung seiner Komponenten beschreibt. Eine Dekomposition der Gesamtfunktion eines Systems in wesentliche Teilfunktionen wird durchgeführt. Diesen Teilfunktionen werden geeignete Wirkprinzipien bzw. Lösungselemente zugeordnet. Durch Verknüpfung der Wirkprinzipien/ Lösungselemente (über Stoff, Energie- und Informationsflüsse) wird die Wirkstruktur erstellt. Unter Berücksichtigung der funktionalen und räumlichen Randbedingungen wird die Wirkstruktur zu einer Baustruktur weiter konkretisiert, damit ein domänenübergreifendes Lösungskonzept erreicht werden kann.

Domänenspezifischer Entwurf

Das domänenübergreifende Lösungskonzept bildet die Basis für weitere Konkretisierung, wobei die weitere Konkretisierung meist getrennt in den beteiligten Domänen abläuft. Die Entwicklung in den einzelnen Domänen erfolgt nach spezifischen Entwicklungsmethodiken der jeweiligen Domänen. Detailliertere Auslegung und Berechnung sind notwendig für die Gewährleistung der Funktionserfüllung.

Systemintegration

Die in den einzelnen Domänen aufgestellten Funktionen der Teilsysteme oder Komponenten werden zu einem übergeordneten Ganzen nach einer bestimmen Art integriert, damit das Zusammenwirken untersucht werden kann. Die Integrationsart ist entsprechend der Aufgabenstellung auszuwählen:

- Integration verteilter Komponenten
- Modulare Integration
- Räumliche Integration

Eigenschaftsabsicherung

Während der Systemintegration findet eine ständige Überprüfung des Entwurfsfortschritts anhand des spezifizierten Lösungskonzepts und der Anforderungen statt. Damit ist es sicherzustellen, dass die realisierten mit den gewünschten Systemeigenschaften übereinstimmen. Dazu bilden zwei Begriffe Verifikation und Validierung die verschiedenen Stufen zur Sicherung der geforderten Systemeigenschaften:

- Verifikation: Prüfung, ob eine Realisierung mit der Spezifikation übereinstimmt.
- Validierung: Prüfung, ob die Spezifikation mit den Anforderungen übereinstimmt.

Modellbildung und –analyse

Wegen der komplexen Struktur und der disziplinübergreifenden Charakter sollten mechatronische Systeme im Rechner abgebildet werden. Mit Hilfe von Modellen und rechnerunterstützten Werkzeugen zur Simulation werden die Abbildung und Untersuchung der Systemeigenschaften gewährleistet, damit die Komplexität mechatronischer Systeme beherrscht werden kann. Daher ist eine modellbasierte Entwicklung gefordert.

Produkt

Das Output des Makrozyklus ist ein Produkt, wobei das Produkt nach Durchlauf eines Zyklus nicht unbedingt den fertigen Zustand erreichen muss. Vielmehr ist eine Reihe von Durchläufen notwendig, um das Produkt zunehmend konkretisieren zu können (Produktreife). Produktreifegrade können z.B. das Labormuster, das Funktionsmuster oder das Vorserienprodukt etc. sein.

2.4.2 Systemspezifikation

Die Anfangsphase der Systementwicklung, also die Phase Anforderung im V-Modell, dient dazu, die Aufgabestellung von dem Entwicklungsauftrag zu beschreiben, d.h. das zu entwickelnde System zu spezifizieren. Diese Phase hat eine entscheidende Bedeutung für die folgenden Phasen, da die Entwicklungsgrundlage hier gebildet werden muss.

Während des Systementwicklungsprozesses kann man die Entwicklungsfehler nach dem Zeitpunkt des Auftretens als fünf Kategorien unterscheiden: Spezifikationsfehler, Entwurfsfehler, Implementierungsfehler, Integrationsfehler und Fertigungsfehler. Darin stellen Spezifikationsfehler falsche, unvollständige oder nicht eindeutige Beschreibungen der Anforderungen und Randbedingungen dar. Diese fehlerhafte Spezifikation führt dann zu den nachfolgenden Fehlern.[52] Aus diesem Grund sollte eine vollständige und fehlerfreie Systemspezifikation zunächst festgelegt werden.

Eine vollständige Systemspezifikation muss vollständige Informationen enthalten. Die Systemspezifikation umfasst folgende Informationen:

- Beschreibung der (Sub-) Systeme,
- Beschreibung der (Haupt-)Funktionen,

[52] Vgl. Eckrich, 1996, S.14

- Beschreibung der Schnittstellen zwischen den (Haupt-)Funktionen, (Sub-)Systemen untereinander,
- Beschreibung von externen Schnittstellen und
- Festlegung von Restriktionen, Annahmen und Abhängigkeiten.[53]

„Vollständig“ bedeutet nicht nur alle Informationen, die zur Beschreibung eines Systems nötig sind, zu beschaffen, sondern auch die Funktionen, die in dem Entwicklungsprozess festgelegt sind, zu erfüllen. Die Systemspezifikation muss folgende Funktionen während des Entwicklungsprozesses erfüllen:

- **Anforderungsanalyse**: Eine wichtige Aufgabe der Systemspezifikation ist es, die oft vagen Anforderungen eines Auftraggebers präzise zu beschreiben. Die geeigneten Spezifikationstechniken sind hier sehr hilfreich, um die Widersprüche, Zweideutigkeiten und Unvollständigkeiten der Beschreibungen zu beseitigen und damit eine ausführbare Spezifikation zu schaffen. Typischerweise ist die Umwandlung vom Lastenheft im Pflichtenheft.
- **Verifikation und Validierung**: Verifikation und Validierung dienen zur Überprüfung der Ergebnisse einer Entwurfsphase. Die Systemspezifikation stellt die wesentliche Grundlage bzw. Kriterien für die Bewertung der Ergebnisse dar.
- **Dokumentation**: Die Spezifikation kann man als ein Medium zur Kommunikation zwischen Auftraggeber und Auftragnehmer, sowie zwischen den einzelnen Entwicklungsteams innerhalb eines Entwicklungsprozesses verstehen. Sie sind rechtverbindliche Entwicklungsgrundlagen, deshalb müssen sie von allen an der Entwicklung Beteiligten abhängig vom gewünschten Grad an Detaillierungstiefe verstanden werden.
- **Systemanalyse und Evaluierung**: Neue Systeme sind oft auf der Basis der bereits bestehenden Systeme entwickelt. Die bereits bestehenden Systemspezifikationen können als Unterstützung zur Analyse der neuen Systeme dienen.
- **Planung**: Die Systemanforderungen sollen nach physikalischen und logischen Aspekten strukturiert werden. Diese plant und organisiert gleichermaßen die Entwicklungsarbeit.[54]

Die Frage ist, wie man die Systemspezifikation vollständig beschreiben kann. Um die Anforderungen an einen Systementwurf in einer Spezifikation vollständig zu erfassen,

[53] Bellalouna, 2009, S.20
[54] Vgl. Eckrich, 1996, S.25

hat Eckrich vier wesentliche Sichtweisen zusammengefasst, unter denen ein zu entwickelndes System betrachtet werden kann. Die vier Sichtweisen sind Verhalten, Struktur, Technologie und Test. Bei der Sicht „Systemverhalten“ handelt es sich um dynamische, d.h. zeitabhängige Vorgänge in einem System. In der Sicht „Systemstruktur“ werden die Modularisierung und die Hierarchisierung des Systems berücksichtigt. Bei der Entwicklung solcher Systeme neben den mechanischen Komponenten stehen die elektronischen Komponenten und Software im Vordergrund. Das Zusammenspiel zwischen diesen Komponenten wird in der Sicht „Systemtechnologie“ beachtet. Und in der Sicht „Systemtest“ wird die Überprüfung des Systems berücksichtigt, so dass das Ergebnis jeder einzelnen Entwurfsphase die Anforderungen aus der vorhergehenden Entwurfsphase erfüllt.[55]

Entsprechend den vorangegangenen Betrachtungen kann das System auf verschiedenen Abstraktionsebenen in einer Systemspezifikation beschrieben werden. Sie sind notwendig und voneinander abhängig. Zunächst sollen die Aufgabe, die Ziele und Nebenbedingungen des zu entwickelnden Systems formuliert werden. Dazu werden die Anforderungen des Systems durch die Anforderungsanalyse ermittelt, strukturiert und geprüft. Das Ergebnis der Anforderungsanalyse ist die Anforderungsspezifikation. Im darauffolgenden Schritt werden die zeitlichen bzw. logischen Abläufe der Systemfunktionen gemäß der Anforderungsspezifikation in der Verhaltensspezifikation beschrieben. Dann werden die Systemkomponenten und ihre Beziehungen sowie die Schnittstellen zwischen dem System und der Umwelt bestimmt. Dadurch wird eine interne Struktur eines Systems dargestellt, die in der Architekturspezifikation festgelegt wird. [56] Anforderungs-, Verhaltens- und Architekturspezifikation sowie ihr Zusammenhang bilden die Systemspezifikation, die das System vollständig beschreiben kann. Für Systems Engineering sind sie aber auch als Bausteine zu betrachten (siehe Kapital 2.3).

Bei der Systemspezifikation spielen die Spezifikationstechniken auch eine wichtige Rolle, da durch diese die für Entwicklung berechtigten Spezifikationen erstellt werden. Die klassischen Beschreibungsformen, die für Beschreibungen verschiedener in der Spezifikationen enthaltenen Informationen eingesetzt werden, sind sprachliche, grafische oder formale Darstellungen, z.B. Fließtext, Verweise, Zahlenwerte, Flussdiagramme, Formeln, Skizzen, usw. Aber durch diese klassischen Beschreibungsformen können die funktionalen Anforderungen nicht mit einer guten

[55] Vgl. Eckrich, 1996, S.26ff
[56] Vgl. Eckrich, 1996, S.29

Qualität beschrieben werden, wobei es zahlreiche Schwachstellen gibt, z.B. widersprüchliche Aussagen, Fehlen wichtiger Informationen, mangelnde Strukturierung, Mehrfacherklärung, oder unrealisierbares Wunschdenken, usw. Diese Schwachstellen führen häufig zu Missverständnissen, Unklarheiten sowie Fehlinterpretationen der zu entwickelnden Aufgaben.[57] Somit braucht man eine eindeutige und widerspruchfreie Spezifikationstechnik, die die funktionalen Anforderungen bzw. Systemverhalten beschreiben kann.

Einsatz von Modellen in der Spezifikation verbessert die Eindeutigkeit und Verständlichkeit der Beschreibung und vermeidet außerdem Spezifikationsfehler. In den vergangenen Jahren sind eine Vielzahl von Spezifikationstechniken zur Verhaltensbeschreibung entwickelt worden, die die dynamischen Vorgänge in einem System auf ein Verhaltensmodell abbilden. Da die Anforderungsspezifikation, die Architekturspezifikation und die Verhaltensspezifikation innerhalb einer Systemspezifikation nötig und voneinander abhängig sind, besteht nun die Frage, ob die modellhafte Beschreibung eines kompletten Systems (Systemanforderungen, –architektur und -verhalten) möglich ist, damit eine noch bessere Spezifikation (Verständnis über den Gesamtwert eines Produkt) erreicht werden kann.

In folgendem Kapitel wird der Ansatz von MBSE vorgestellt, der die Möglichkeit zur modellhaften Beschreibung eines kompletten mechatronischen Systems in der Spezifikationsphase bietet und damit eine Reihe von Vorteilen aufweist.

2.5 Modellbasiertes Systems Engineering

Traditionelles Systems Engineering ist eine dokumentenbasierte Methode, die durch die Generierung von textuellen Spezifikationen charakterisiert wird. Das heißt, dass die Spezifikationen von verschiedenen Entwicklungsbereichen in Form von Dokumenten verfasst werden, die als Grundlage für die nachgeschalteten Entwicklungsabteilungen dienen. Diese Entwicklung läuft so ab, dass die Systemanforderungen zunächst festgelegt werden. Aus diesen wird dann eine Systemarchitektur abgeleitet. Die Anforderungs- und Architekturspezifikation wird als Eingangsprodukt in die einzelnen Teilentwicklungsbereiche Mechanik, Informatik und Elektronik übertragen. Dann werden die disziplinspezifischen Spezifikationen in den einzelnen Teilentwicklungsbereichen erstellt und aufgrund dieser Spezifikationen

[57] Vgl. Eckrich, 1996, S.17

werden Systemkomponenten entwickelt.[58] Bei dem dokumentenbasierten Systems Engineering (DBSE) werden die Systeme dokumentenzentriert entwickelt, es entsteht kein kohärentes Modell des Gesamtsystems.

DBSE scheint eine ganzheitliche Methode zu sein, aber sie hat einige grundlegende Schwachstellen. Die Informationen, die tief in Texten versteckt sind, sind auf dem ersten Blick nicht offensichtlich. Weiterhin sind die Informationen in diversen Dokumenten verteilt, die unterschiedlich Aspekte des Systems beschreiben. Das schadet der Vollständigkeit, der Konsistenz und der Nachvollziehbarkeit von Beziehungen zwischen den Dokumenten. Die systemspezifischen Aspekte sind demzufolge schwer zu verstehen und zu analysieren, und die Rückverfolgbarkeit ist auch schwer zu gewährleisten. Dies führt zu einer schlechten Synchronisation zwischen der Konzeptionsphase und Entwurfsphase. Die Wiederverwendung der bereits entstehenden Systemspezifikation für eine Weiterentwicklung (Systemanalyse und Evaluierung) wird erschwert.[59] Unter anderen wurde im letzten Kapitel erwähnt, dass die textuelle Spezifikation die Qualität der Anforderungen nicht hinreichend reflektieren kann und damit zur Gefahr einer fehlerhaften Entwicklung führt.

Um diese Schwachstellen zu beseitigen und die interdisziplinären technischen Systeme noch besser zu entwickeln, wurde ein neuer Ansatz von INCOSE vorgeschlagen, – das modellbasierte Systems Engineering (MBSE). Die Definition des MBSE nach INCOSE lautet:

„Model Based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [60]

Im deutschsprachigen Raum hat man versucht, das MBSE noch vollständig und konkret bzw. noch besser zu definieren. In einem Forum wurden folgende Definitionen von MBSE zusammengefasst:

„Das MBSE führt von der reinen dokumentenbasierten Entwicklung hin zu einem geschlossenen, in sich konsistenten Systemmodell, um das zu entwickelnde System besser zu verstehen, analysieren, detaillieren und kommunizieren, und weiterzuleiten mit dem Ziel, die Qualität der Spezifikations- und Testartefakte und damit des zu

[58] Vgl. Alt, 2012, S.1
[59] Vgl. Friedenthal; More; Steiner, 2012, S.16
[60] INCOSE, 2007, S.23

entwickelnden Systems zu steigern, Entwicklungskosten zu reduzieren und time-to-market zu beschleunigen.“ [61]

oder

„MBSE ist der Entwurf, die Spezifikation und die Verifikation und Validierung von komplexen Systemen mit Hilfe von Modellen, wobei jedes einzelne Modell eine durch Abstraktion erzeugte. Repräsentation der statischen Struktur und des Verhaltens des zu entwickelnden Systems und ggf. seiner Umgebung bzw. Testumgebung ist.“ [62]

Im Gegensatz zu DBSE steht beim MBSE ein sogenanntes Systemmodell im Mittelpunkt. Die während der Entwicklung entstehenden Informationen werden zentral in diesem Modell verwaltet und gepflegt. Das erleichtert den Informationsaustausch zwischen den Entwicklungsingenieuren, die in verschiedenen Domänen arbeiten. Die einzelnen Entwicklungsingenieure können die Informationen, die für ihre Sicht wichtig sind, aus dem Modell herausnehmen, und gleichermaßen die neuen Informationen dem Modell hinzufügen. Somit gibt es keinen Medienbruch mehr zwischen den Entwicklungsingenieure.[63]

Das Systemmodell kann aus verschiedenen Perspektiven betrachtet werden, die den Aspekten des Systems entsprechen. Vollständigkeit, Konsistenz und Beziehungen zwischen verschiedenen Aspekten des Systems sind dadurch sichergestellt. Weiterhin sind die Auswirkungen der Veränderungen dadurch leichter zu analysieren.

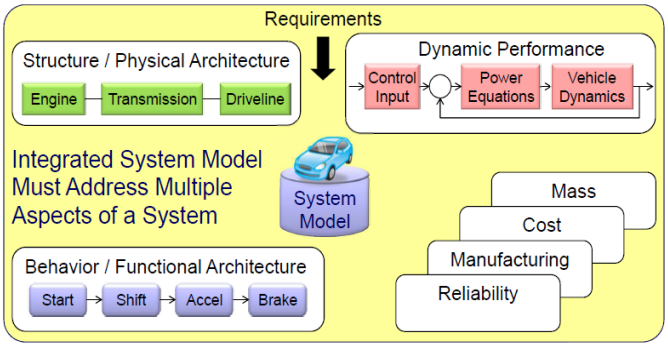

Abbildung 10: System Model [64]

[61] http://www.xing.com/net/syng/modellbasiertes-systems-engineering-475034/was-ist-mbse-33792101/p10 (09.04.2013)

[62] http://www.xing.com/net/syng/modellbasiertes-systems-engineering-475034/was-ist-mbse-33792101/p10 (09.04.2013)

[63] Vgl. Alt, 2012, S.1

[64] Chris Paredis: Why Model-Based Systems Engineering? Benefits and Payoffs, (2008-2011) http://vpe.mv.uni-kl.de/cms/fileadmin/user_upload/PDF/PLM_Future_2012/02_PLM-Future-2012_Paredis.pdf (09.04.2013)

Im MBSE ermöglichen die systemspezifischen Modellierungssprachen die Systeme modellhaft auf einer allgemeinen Ebene zu spezifizieren. z.B. mit SysML werden Systemanforderungen, -struktur und –verhalten und ihre Beziehungen in einem Systemmodell modelliert, wodurch eine Gesamtansicht von dem zu entwickelnden System möglichst früh dargestellt wird. Diese modellhafte Systemspezifikation ist für alle an der Entwicklung beteiligten Disziplinen verständlich. So ist MBSE ein interdisziplinärer Ansatz, der den Entwicklungsingenieuren hilft, den Überblick über komplexe Systeme zu behalten, den Zusammenhang zu verstehen und die Spezifikation und damit alle definierten Anforderungen zu erfüllen.[65] Es erlaubt die Kombination verschiedener Disziplinen und erleichtert die Erfassung der Komplexität des Systems.

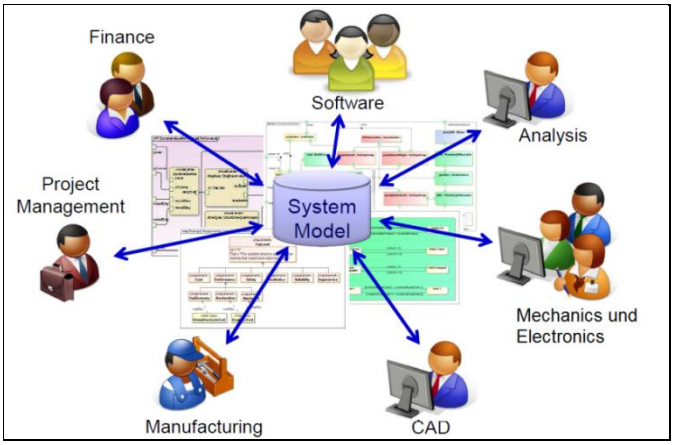

Abbildung 11: Model based Systems Engineering [66]

Zusammenfassung der Vorteile von MBSE im Vergleich mit DBSE:

- Verbesserung der Kommunikation zwischen den am Entwicklungsprozess beteiligten Stakeholders.
- Verbesserte Fähigkeit zum Erfassen der Komplexität des Systems durch das Systemmodell.
- Verbesserte Produktqualität durch eine eindeutige und präzise System-spezifikation.
- Verbesserte Wiederverwendbarkeit der Informationen für neue Entwicklung.

[65] Vgl. Eigner; Gilz; Zafirov: Interdisziplinäre Produktentwicklung – Modellbasiertes Systems Engineering. http://www.plmportal.org/forschung-details/items/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (10.04.2013)

[66] Vgl. Chris Paredis: Why Model-Based Systems Engineering? Benefits and Payoffs, http://vpe.mv.uni-kl.de/cms/fileadmin/user_upload/PDF/PLM_Future_2012/02_PLM-Future-2012_Paredis.pdf (10.04.2013)

- Verbesserte Fähigkeit zum Lehren und Lernen der Grundlagen des SE durch eine klare und eindeutige Repräsentation vom Konzept.[67]
- Verringerung der Risiken in der Entwicklung.
- Verringerung der Entwicklungszeit und Reduzierung der Entwicklungskosten.

Deshalb wird die Modellbasierte Methode als Ersatz der dokumentenbasierten Methode im SE-Prozess integriert und in der Praxis verwendet.

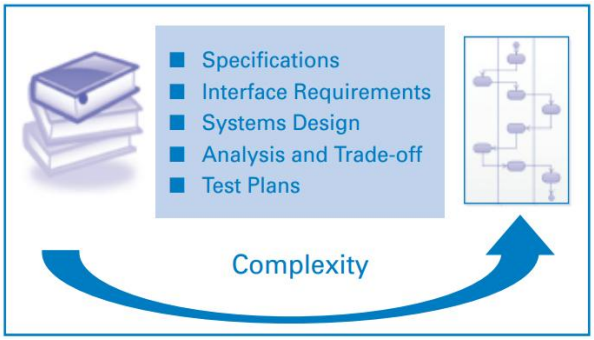

Abbildung 12: Moving from document centric to model centric [68]

2.5.1 Modellbasierte Entwicklung

In der modellbasierten Entwicklung spielt das Modell eine zentrale Rolle, aus dem dann die gewünschten Zwischen- oder Endprodukte (Artefakte), z.B. Quellcode, Dokumentation oder weitere spezifische Modelle weitgehend automatisiert abgeleitet werden sollen. [69] Das Prinzip der Modellierung wird als Grundidee der modellbasierten Entwicklung angewendet. Und Model-Driven Architecture, die ein wichtiger Standard von Object Management Group (OMG) ist, zeigt darauf, wie modellbasierte Entwicklung funktioniert.

Modell

Das internationale Standardisierungsgremium OMG definiert das Modell als die Repräsentation eines oder mehrerer Konzepte, die physikalisch umgesetzt werden können. In der Regel wird hierdurch ein bestimmter Interessenbereich beschrieben.[70] Das Modell wird unter Verwendung verschiedener Darstellungsformen grafisch

[67] Vgl. Friedenthal; Griego; Sampson: INCOSE Model Based Systems Engineering (MBSE) Initiative, (2007). http://www.incose.org/enchantment/docs/07docs/07jul_4mbseroadmap.pdf (12.04.2013)
[68] http://www.incoseonline.org.uk/Program_Files/Publications/zGuides_9.aspx?CatID=Publications (12.04.2013)
[69] Vgl. Alt, 2012, S.19
[70] Vgl. Maurer; Schultze, 2010, S121

dargestellt und oftmals als Kombination von Text und grafischen Symbolen präsentiert.

Ein wesentliches Merkmal eines Modells ist, dass es eine abstrakte Darstellung ist, die nicht alle Details der modellierten Einheit innenhalb des Interessenbereiches enthält. [71] Diese Abstraktion bietet die Möglichkeit, die Artefakte aus einer Abstraktionsstufe in eine weitere zu transformieren. Aus einem Modell kann ein zweites Modell durch Hinzufügen detailliertere Informationen auf einer niedrigen Abstraktionsebene generiert werden. Und das Modell kann aus unterschiedlichen Perspektiven betrachtet werden.

Model-Driven Architecture

Model-Driven Architecture (MDA) ist eigentlich eine Methodik für die Softwareentwicklung, die von dem OMG an 2001 eingeführt wurde. MDA ist auf Basis von vielen wichtigen OMG standardisierten Hilfswerkzeugen wie Unified Modeling Language (UML), Meta Object Facility (MOF) etc. aufgebaut. Als ein architektonisches Framework dient MDA zur Visualisierung, Speicherung und dem Austausch von Artefakt-Modellen，die während der modellbasierten Software-entwicklung entstehen. Diese werden miteinander in Beziehung gesetzt.[72]

Mit der Modellierungssprache UML bildet MDA abstrahierte und maschinenlesbare Modelle, welche unabhängig von der Implementierungstechnik sind. Somit bleibt MDA technisch abstrakt.[73] Aufgrund des Abstraktionsprinzips (Transformation) wird das Grundkonzept der MDA dargestellt. Und es geht darum, dass ein Modell im Mittelpunkt der Entwicklung steht, aus dem neue Modelle mit Hilfe von einer Modelltransformation erzeugt werden. Das zeigt darauf, wie modellbasierte Entwicklung funktioniert: Die Entwicklungsinformationen liegen in einem vom Rechner verarbeitbaren Modell, das als primäres Artefakt in dem Mittelpunkt der Entwicklung steht. Mit Hilfe von Werkzeugen sollen die Informationen in anderen Formen abgeleitet werden.[74]

Neben dem Grundkonzept spezifiziert MDA drei verschiedene Gesichtspunkte auf das zu entwickelnde System, und zwar Computation Independent, Platform Independent und Platform Specific Viewpoints. Für jedes dieser Gesichtspunkte wird

[71] Vgl. Friedenthal; More; Steiner, 2012, S.21
[72] Vgl. Maurer; Schultze, 2010, S124
[73] Vgl. Alt, 2012, S.22
[74] Vgl. Alt, 2012, S.22f

ein voreingestelltes Modell definiert. Es ist besser, diese Modelle als Abstraktionsebene zu verstehen, da in jeder dieser drei Ebenen eine Reihe von Modellen dargestellt werden kann.[75]

- Computation Independent Model (CIM): Es ist ein Modell, das keine Details über die Struktur des Systems enthält, sondern nur die Anforderungen und Rahmenbedingungen, die von der Ausführung unabhängig sind. Es wird auch als Kontextmodell genannt, da es eine bessere Verständigung zwischen Experten aus verschiedenen Domänen und denjenigen Entwicklern, die für technische Realisierung des Systems zuständig sind, fördert.
- Platform Independent Model (PIM): Es beschreibt die Systemstruktur des zu entwickelnde Systems auf einer realisierungsunabhängigen Ebene. Es zeigt einen ausreichenden Grad an Unabhängigkeit, damit die aus diesem mit entsprechenden Transformationen erzeugte Zielmodelle den verschiedenen technologiespezifischen Plattformen zugeordnet werden können. Diese Zielmodelle haben die gleiche Funktionalität.
- Platform Specific Model (PSM): Es ist ein Zielmodell, welche aus dem PIM mit bestimmter Transformation auf einer ausgewählten technologiespezifischen Plattform erzeugt wird. Es kann aber auch ein Zwischenmodell sein, wenn es nicht alle zur Realisierung notwendige Daten enthält.[76]

Folgende Abbildung zeigt den Entwicklungsprozess einer Software durch Anwendung der Model-Driven Architecture, wobei die Artefakt-Modelle miteinander in Beziehung gesetzt werden. Es läuft von einem plattformunabhängigen Spezifikationsmodell bis hin zu einem plattformspezifischen Modell ab.

[75] Vgl. Frank Truyen, S.5
[76] Vgl. Frank Truyen, S.5

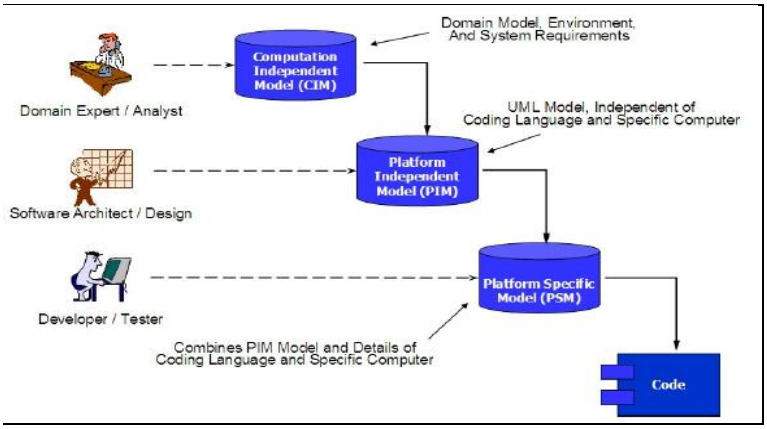

Abbildung 13: Model-Driven Architecture [77]

Cloutiere hat den MDA Ansatz in Systems Engineering angewandt und er weist darauf hin, dass die Effizienz im SE-Projekt durch Anwendung der MDA um 10-20% verbessert werden kann, wenn diese Methodik für SE adaptiert ist.[78] Eine grafische Darstellung der Anwendung der MDA in SE wird in folgender Abbildung gezeigt.

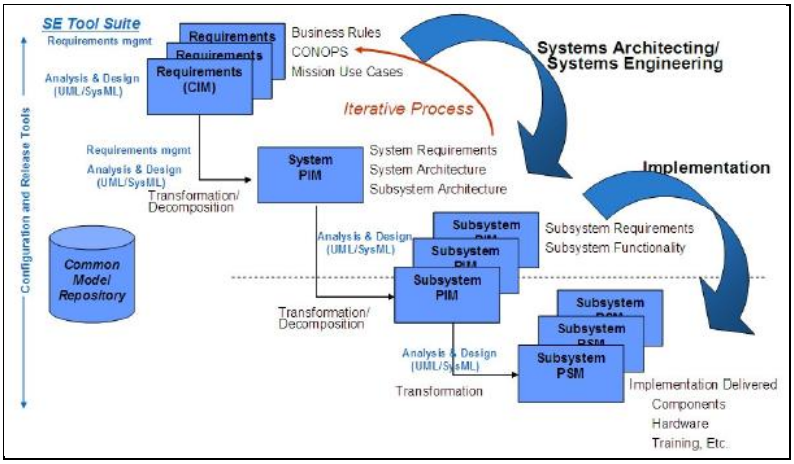

Abbildung 14: Anwendung der MDA in System Engineering [79]

Durch Anwendung der MDA werden die Systeme modellbasiert entwickelt. Diese Abbildung stellt den modellbasierten Entwicklungsprozess dar, wobei die verschiedenen Artefakte den entsprechenden Gesichtspunkten zugeordnet sind. Common Model Repository, configuraton und Release tools, Requirements Management und die Modellierungssprache UML und SysML sind die Werkzeuge für modellbasierte Entwicklung und bilden hier eine SE Tool Suite. CIM umfasst die Systemanforderungen, System-stakeholder Bedürfnisse und Geschäftsregeln, die das zu entwickelnde System beeinflussen. Die nächste Abstraktionsstufe PIM

[77] Estefan, 2008, s.54
[78] Vgl.Estefan, 2008, s.55
[79] Estefan, 2008, s.55

spezifiziert die Systemanforderungen (Pflichtenheft), die aus den Kundenanforderungen (Lastenheft) übergesetzt werden, und die Systemarchitektur sowie das Systemverhalten. Hier ist ein Multi-Level PIM notwendig, um die Komponenten eines Systems zu spezifizieren. Die Subsystem-PIMs können aus dem System-PIM abgeleitet werden. Sie beschreiben die Anforderungen und die Funktionalitäten der Systemkomponenten. Aus diesen Spezifikationsmodellen werden dann PSMs der Systemkomponenten abgeleitet. Sie beschreiben die logischen, physikalischen Systemkomponenten, und dienen als Lösungskonzept zur Implementierung.

CIM ist nur ein Kontextmodell, das die externen Anforderungen repräsentiert. PIM ist die Spezifikation eines Systems, die die Systemanforderungen, -architektur und -verhalten umfasst. Und PSM ist die Komponentenspezifikation, die die Komponente konkret spezifiziert. Diese Drei Modelle können mit dem UML/SysML erstellt werden und bilden zusammen das Systemmodell ab.

2.5.2 Das Systemmodell

Das Systemmodell spielt eine zentrale Rolle als Systemspezifikation für Systementwicklung. Im Rahmen von MBSE hat Weilkiens das Systemmodell so definiert:

„Das Systemmodell im Kontext des MBSE ist das Abbild eines realen oder noch zu entwickelnden Systems, wobei mittels Abstraktion nur die für einen definierten Zweck relevanten Attribute berücksichtigt werden"[80]

Das Systemmodell wird mittels eines Modellierungswerkzeugs erzeugt und in einem sogenannten „Model Repository" abgelegt.[81] Das Systemmodell ergibt sich aus dem Verknüpfen der Modellelemente, die in Form von Diagrammen durch grafische Symbolen dargestellt und in einem Model Repository gespeichert werden. Das Modellierungswerkzeug dient zum Darstellen, Löschen und Ändern der einzelnen Modellelemente und ihrer Beziehungen. In Diagrammen werden die grafischen

[80] Weilkiens: Zukunftsdisziplin Modellbasiertes Systems Engineering (19.-20.05.2011) http://www.conimit.de/uploads/tx_vitramemberadmin/literature/Zukunftsdisziplin-Systems-Engineering.weilkiens.oose.2011.pdf (13.04.2013)

[81] Vgl. Friedenthal; More; Steiner, 2012, S.17

Symbole als Informationsträger verwendet, um die Informationen in dem Modell einzugeben oder die Informationen aus dem Modell auszulesen.[82]

Das Systemmodell enthält die Anforderungs- Design-, Analyse-, und Testinformationen, die als Modellelemente miteinander in Beziehungen stehen. Durch diese Beziehungen wird die Rückverfolgbarkeit (Traceability) gewährleistet.[83] Alle Aspekte des Systems werden in dem Systemmodell durch Modellelemente und Beziehungen zwischen den Modellelementen festgehalten. Abbildung 15 zeigt ein mit der Modellierungssprache SysML beschriebenes Systemmodell. Es ist eine Verknüpfung von den Modellelementen „Requirements“, „Structure“, „Behavior“ und „Parametrics“, die die vier Hauptaspekte des Systems repräsentieren.

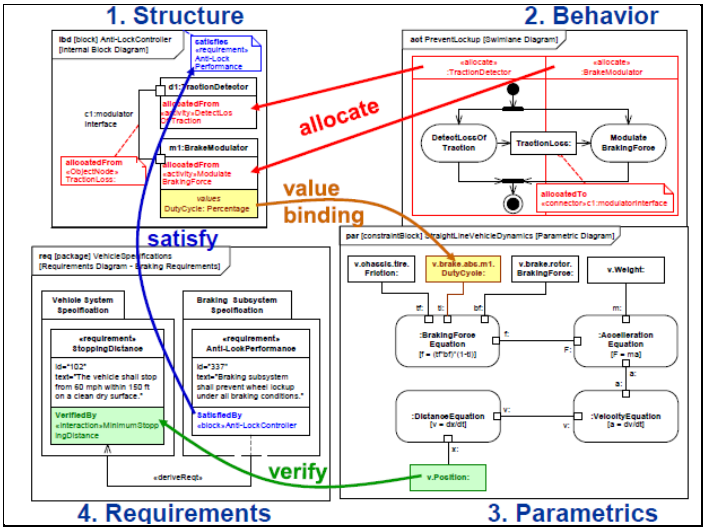

Abbildung 15: Systemmodell mit SysML [84]

Systems Engineering sollte die Teilsysteme zu einem kohärenten Ganzen zusammenbauen. MBSE muss diesen grundlegenden Schwerpunkt unterstützen. Das Systemmodell steht im Mittelpunkt und unterstützt Aktivitäten jeder Entwicklungsphase. Deshalb kann das Systemmodell als ein Integration-Framework für andere Modelle und Artefakte betrachtet werden, die für verschiedene Disziplinen erzeugt werden. Diese Modelle und Artefakte umfassen die Systemdokumente/Textspezifikation, spezifische Modelle, Analysemodelle und Testmodelle etc.[85] Sie sind lediglich Sichten auf das Systemmodell, die aus dem

[82] Vgl. Friedenthal; More; Steiner, 2012, S.19

[83] Vgl. Friedenthal; More; Steiner, 2012, S.19

[84] Vgl. Chris Paredis: Why Model-Based Systems Engineering? Benefits and Payoffs, http://vpe.mv.uni-kl.de/cms/fileadmin/user_upload/PDF/PLM_Future_2012/02_PLM-Future-2012_Paredis.pdf (10.04.2013)

[85] Vgl. Friedenthal; More; Steiner, 2012, S.523

Systemmodell generiert, bzw. durch Modelltransformation erzeugt werden.[86] Alle Modelle werden in dem Model Repository gespeichert und verwaltet. (siehe Abbildung 16)

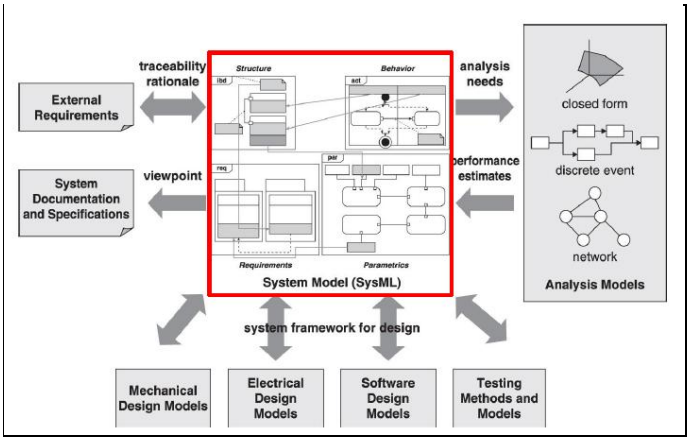

Abbildung 16: Systemmodell als ein Integration-Framework [87]

2.5.3 Modellierungssprache SysML

Das Systemmodell dient im MBSE als ein Integration-Framework, das alle disziplinspezifischen Modelle integrieren muss. Eine standardisierte Modellierungssprache ist erforderlich, um ein interdisziplinäres Systemmodell darzustellen, das das Systems Engineering durchgängig und disziplinübergreifend unterstützen kann. Die SysML ist genau eine standardisierte Modellierungssprache für die Spezifikation, die Analyse, das Design, und die Verifikation und Validierung von Systemen und deren Systemelemente wie beispielsweise Software, Hardware, Informationen, Prozesse, Personen und Gegenstände.[88]

Die SysML wurde auf der Basis von UML, welche zur Modellierung der Softwaresysteme dient, von OMG und INCOSE entwickelt. Sie benutzt einen großen Teil von UML und bietet zusätzlich auch neue Modellierungs- diagramme, um komplexe Systeme vollständig zu modellieren.[89] Mit SysML können folgende Aspekte des Systems, Komponente, und Objekte dargestellt werden:

---

[86] http://www.xing.com/net/syng/modellbasiertes-systems-engineering-475034/was-ist-mbse-33792101/p0 (22.04.2013)

[87] Sanford Friedenthal: Model-Based Systems Engineering (MBSE) 2012. http://www.lotustech.co.kr/PHX_forum2012/MC/SystemEngineering/Model-based%20Systems%20Engineering%20with%20SysML_Friedenthal.pdf (22.04.2013)

[88] Vgl. Hausding, 2010, S.7

[89] Vgl. Hausding, 2010, S.7

- Strukturelle Komposition, Querverbindung, und Klassifikation
- Funktionsbasierte, Sequenzbasierte, und Zustandsbasierte Verhalten
- Einschränkungen auf die physikalischen und leistungsfähigen Eigenschaften
- Allokation zwischen Verhalten, Strukturen, und Einschränkungen
- Anforderungen und die Beziehungen zu anderen Anforderungen, Design-elementen, und Testfälle [90]

Mit der SysML können also die Anforderungen, die Struktur und das Verhalten eines Systems, die die Bausteine des SE sind, auf formale Art und Weise beschrieben und miteinander in Beziehung gesetzt werden. Dadurch wird ein Systemmodell erzeugt, welche als Entwicklungsgrundlage für die einzelnen Entwicklungsdisziplinen dient. Die Standardisierung der SysML ermöglicht es, ein komplexes System auf einer allgemeinen Ebene zu spezifizieren. Alle Beteiligten, die aus verschiedenen Domänen sind, können deshalb das System bzw. die Systemzusammenhänge leicht und gleichermaßen verstehen und kommunizieren.[91] So spielt SysML eine sehr wichtige Rolle als „Enabler“ für MBSE.

Für die Modellierung benutzt SysML verschiedene Diagramme. Die Abbildung 17 gibt einen Überblick von den in SysML definierten Diagrammen. Sie werden in drei Kategorien aufgeteilt, die den drei Bausteinen des SE entsprechen. Das Anforderungsdiagramm dient zur Modellierung der Systemanforderungen, die strukturellen Diagramme dienen zu Modellierung der Systemstruktur, und die Verhaltensdiagramme dienen zur Modellierung des Systemverhaltens.

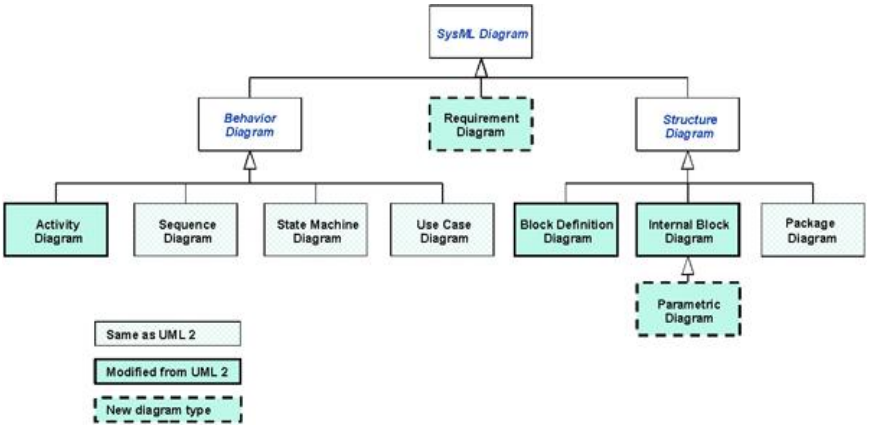

Abbildung 17: Diagrammen in SysML [92]

[90] Friedenthal; More; Steiner, 2012, S.29

[91] Vgl. Alt, 2012, S.30

[92] http://www.omgsysml.org/ (23.04.2013)

- Anforderungsdiagramm: Mit dem werden die textbasierten Anforderungen und ihre Beziehungen mit anderen Anforderungen, Designelemente und Testfälle dargestellt.
- Aktivitätsdiagramm: Mit dem werden die Abläufe der Aktivitäten und ihre Ein- und Ausgangsdaten sowie der Kontrollfluss zwischen den Aktivitäten dargestellt.
- Sequenzdiagramm: Es stellt die Interaktion dar und spezifiziert den Austausch der Nachrichten zwischen den Teilen eines Systems.
- Zustandsdiagramm: Mit dem werden die Zustände und die Zustandsübergänge dargestellt, die aufgrund von einem System oder einem Teil des Systems in Reaktion auf die Ereignisse stattfinden.
- Anwendungsfalldiagramm: Es stellt die Anwendungsfälle, die Akteure und die Beziehungen zwischen Akteure und Anwendungsfälle dar, und gibt an, was man mit einem System machen will und welche Anwendungsfälle es gibt.
- Blockdefinitionsdiagramm: Es definiert die Systembausteine, die die Teile einer Struktur eines Systems beschreibt. Ihre Komposition und Klassifikation werden dargestellt
- Internes Blockdiagramm: Es stellt die Verbindungen und Schnittstellen zwischen den Bausteinen dar, und gibt an, wie die Bausteine miteinander verbunden werden und welche Informationen zwischen ihnen ausgetauscht werden.
- Paketdiagramm: Es dient zur Organisation des Modells
- Zusicherungsdiagramm: Es definiert die parametrischen Beziehungen zwischen den Eigenschaften der Systembausteine.

Ein Diagramm spezifiziert einen spezifischen Aspekt des Systemmodells. SysML umfasst verschiedene Allokationsbeziehungen zwischen den Systemelementen, die die Zuordnung der Funktionen zu Komponenten, der logischen zu physikalischen Komponenten, und der Software zur Hardware ermöglichen. [93] Durch die Allokationsbeziehungen, die parametrischen Beziehungen und die im Anforderungsdiagramm definierten Beziehungen werden die verschiedenen Systemelemente miteinander verbunden. So ermöglichen diese Beziehungen das Systemmodell aus verschiedenen Perspektiven zu betrachten. Ein einfaches Beispiel ist ein mit SysML modelliertes Systemmodell, das in Abbildung 15 gezeigt wird. Später in Kapitel 5 wird SysML zur Modellierung eines konkreten Systems verwendet.

[93] Vgl. http://www.omgsysml.org/ (23.04.2013)

2.5.4 Erweitertes V-Modell für MBSE

In Kapitel 2.4.1 wurde das V-Modell für Entwicklung mechantronischer Systeme vorgestellt. Durch Einsatz der Methoden aus dem MBSE wurde das V-Modell erweitert. Dieses erweiterte V-Modell wird besonders durch den sogenannten RFLP-Ansatz charakterisiert und dient als Vorgehensweise in MBSE. RFLP (Deutsch: AFLP) ist die Abkürzung für „Requirements“, „Function“, „Logical“ und „Physical“. Sie definiert das Produkt aus vier verschiedenen Sichten und bildet die linke Seite des V-Modells ab. (siehe Abbildung 18)

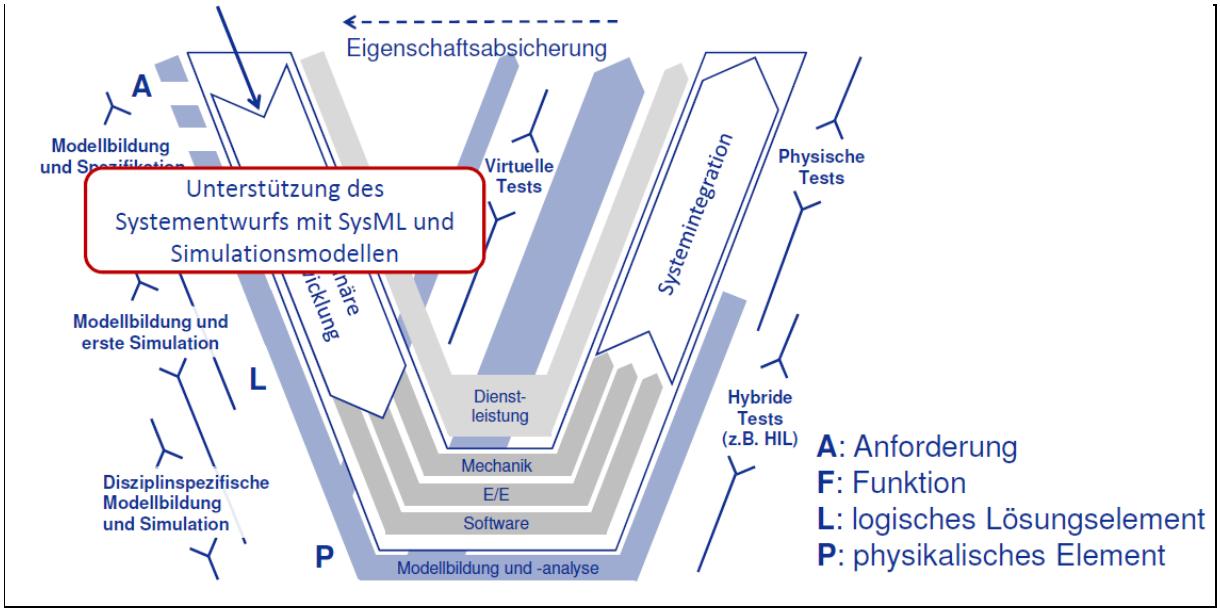

Abbildung 18: Erweitertes V-Modell für MBSE [94]

Das V-Modell wird in zwei Abschnitten untergeteilt. Die linke Seite des V-Modells ist die interdisziplinäre Systementwicklung, wobei ein System sich von abstrakter Spezifikation bis hin zu konkretem physikalischen Entwurf entwickeln lässt. Und die rechte Seite ist Systemintegration, wobei die aufgestellte Subsysteme oder Systemkomponenten zum Ganzen integriert werden, und die Produkteigenschaften übergeprüft werden. Der Unterscheid zwischen diesem und dem in oben beschriebenen V-Modell ist, dass das System in diesem erweiterten Modell nach einer RFLP-Reihenfolge entwickelt werden soll.

In dem erweiterten V-Modell hat man drei Abstraktionseben für die interdisziplinäre Systementwicklung definiert. Dies sind „Modellbildung und Spezifikation“,

[94] Eigner; Gilz; Zafirov: Interdisziplinäre Produktentwicklung – Modellbasiertes Systems Engineering. http://www.plmportal.org/forschung-details/items/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (28.04.2013)

„Modellbildung und erste Simulation“ und „Disziplinspezifische Modellbildung und Simulation“. Auf der Ebene „Modellbildung und Spezifikation“ wird das zu entwickelnde System spezifiziert, d.h. das Systemmodell wird dargestellt. Sie umfasst die erste Entwicklungsphase „R“ (Deutsch ist „A“ die Anforderung) und die zweite Phase „F". Auf der Ebene „„Modellbildung und erste Simulation“ werden logische, simulierbare Modelle dargestellt, die mehrere Disziplinen miteinbeziehen. Diese Ebene entspricht der dritten Entwicklungsphase „L“. Auf der letzten Ebene „Disziplinspezifische Modellbildung und Simulation“ werden die geometrischen Modelle also die CAE-Modelle darstellt, die disziplinspezifisch sind. Diese Ebene entspricht der vierten Entwicklungsphase „P“.[95] RFLP stellt einen fortlaufenden, modellbasierten Lösungsweg von der Anforderung bis hin zu der physikalischen Lösung dar.

RFLP-Ansatz :

- Requirements: Die Entwicklung beginnt mit der Anforderungsdefinition. Die Bedürfnisse der Stakeholders werden identifiziert und als externe Anforderungen definiert. Die externen Anforderungen werden dann durch Anforderungsanalyse auf interne Systemanforderungen übergesetzt. Ein Anforderungsmodell wird als Output dieser Phase mit Hilfe der Modellierungssprache SysML dargestellt.
- Function: Die Systemfunktionen werden definiert, um die Anforderungen zu erfüllen. Eine funktionale Spezifikation soll in der frühen Entwicklungsphase erstellt werden. Die Systemstruktur und das Systemverhalten können auch mit SysML modelliert werden und bilden zusammen mit dem Anforderungsmodell das Systemmodell, das als eine interdisziplinäre und lösungsneutrale Spezifikation dient.
- Logical: Systemstruktur und –verhalten werden in dieser Phase objektorientiert modelliert. Sie werden durch die Definition der logischen Komponenten beschrieben, die wiederum die in der Spezifikation beschriebenen Funktionen realisieren können. Diese logische und physikalische Struktur und Verhalten bilden ein Lösungskonzept für die Realisierung. Die objektorientierten Modellierungs- sprachen z.B. werden hier Dymola oder Matlabs/Simulink zur Modellierung verwendet.

[95] Vgl. Eigner; Gilz; Zafirov: Interdisziplinäre Produktentwicklung – Modellbasiertes Systems Engineering. http://www.plmportal.org/forschung-details/items/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (28.04.2013)

- Physical: Aufgrund des Lösungskonzepts kann die disziplinspezifische Entwicklung stattfinden. Disziplinspezifische Systemkomponenten werden geometrisch entworfen. CAD-System oder disziplinspezifische Berechnung- und Simulationssoftware dienen hier als Autorenwerkzeuge.[96]

Die simulierten Systemkomponenten werden zu einem übergeordneten Ganzen zusammengesetzt. Das Ergebnis wird dann anhand des Lösungskonzepts und Anforderungsmodells übergeprüft. Der ganze Entwicklungsprozess läuft auf Basis der Modelle ab.

Die interdisziplinäre Systementwicklung erfolgt durch die Unterstützungen von zahlreichen computergestützten Werkzeugen. Jede Phase der interdisziplinären Systementwicklung (R, F, L oder P) braucht ein oder mehrere spezifische Autorenwerkzeuge, um bestimmte Modelle zu erstellen und zu verwalten. Die Werkzeuge wie z.B. die CAE-tools für Systemanalyse, CAT-tools für Verifikation und Validation usw. werden ebenfalls gebraucht. Es stellt sich die Frage, ob alle Werkzeuge in einem IT-Werkzeug integriert werden können, sodass es den modellbasierten Entwicklungsprozess durchgängig unterstützen kann.

Im nächsten Kapitel wird das neu definierte CAD-System CATIA V6 vorgestellt, welche auf eine einheitliche, vernetzte PLM-Plattform basiert. Mit CATIA V6 können nicht nur die geometrischen Modelle darstellt werden, sondern auch die Systemmodelle und logischen Modelle. Es bietet einen RFLP-Ansatz zur Entwicklung mechatronischer Produkte.

[96] Vgl. Eigner; Gilz; Zafirov: Interdisziplinäre Produktentwicklung – Modellbasiertes Systems Engineering. http://www.plmportal.org/forschung-details/items/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (28.04.2013)

3 SE mit CAD System - CATIA V6

3.1 Überblick von CATIA V6

Computer Graphics-Aided Three-Dimensional Interactive Application (CATIA) ist ein von Dassault Systèmes entwickeltes CAD/CAM/CAE/PLM-System. Es liefert eine einheitliche Plattform zum Entwurf, Analyse, und Fertigung eines Produkts, um die Produktentwicklung zu vereinfachen und die Entwicklungskosten und –zeit zu reduzieren.

CATIA Version 6 (V6) wurde in 2008 veröffentlicht und umfasst neben der rein physischen Produktdefinition und digitalen Modellierung (digital mock-up) auch die funktionale Produktdarstellung (functional mock-up) unter Berücksichtigung der verschiedenen Ansichten für die Produktentwicklung (Anforderung, funktionale, logische und physische Visualisierung).[97] CATIA V6 ist nicht nur ein 3D-CAD-System, sondern dient als eine IT-Lösung zur Unterstützung des Systems Engineering. V6 PLM-System liefert eine einheitliche Onlineumgebung, die eine Echtzeit Collaboration ermöglicht. Die Beteiligten der Entwicklung, die an entfernten Standorten sind, können durch Internet in Echtzeit zusammenarbeiten. CATIA V6 stellt die Werkzeuge zum co-design (collaborative design) und co-review (collaborative review) des Produkts bereit. Mit diesen Werkzeugen können die Benutzer brainstormen, um die Entwurfs- und Überprüfungsprobleme zusammen lösen zu können.

V6 bietet einen einheitlichen SE-Ansatz zur modellhaften Entwicklung mechantronischer Produkte, wobei ein neues Produkt von der Spezifikation bis hin zu der Design und Simulation durch verschiedene Phasen entwickelt wird. CATIA V6 reduziert die Komplexität der Entwicklung durch die Integration verschiedener Produktentwicklungsansätze in einer einheitlichen Plattform. Es erlaubt die Informationen, die in verschiedenen Entwicklungsphasen entstehen, zentral gespeichert und verwaltet zu werden und stellt damit jeder Benutzer die notwendigen und aktuellen Informationen zur Verfügung.

Dabei stellt CATIA V6 sechs Funktionsmodule für effektive und effiziente Konstruktion bereit. Diese sind:[98]

---

[97] Vgl. http://www.3ds.com/de/products/catia/portfolio/catia-version-6/overview/ (07.05.2013)

[98] Vgl. https://www.cenit.de/de/plm/lsungenprodukte/catia/catia-v6.html (14.05.2013)

- CATIA Analyse: Durch CATIA Analyse lassen sich realistische Simulationen und virtuelle Prüfungen des Produktverhaltens umsetzten.
- CATIA Shape: CATIA Shape bietet eine produktive Umgebung zur Gestaltung aller Art von Formen, Freiformflächen und Design-Oberflächen.
- CATIA Mechanical: Mit CATIA Mechanical können 3D Bauteilen (z.B. Gussteile, Schmiedeteile, Blechbauteile, Composite-Strukturen, mechanische Baugruppen, ect.) einschließlich der fertigungsspezifischen Spezifikationen und funktionalen Toleranzen erstellt werden.
- CATIA Equipment: CATIA Equipement bietet eine integrierte Umgebung zum Entwurf von elektronischen, elektrischen und fluidischen Systeme im Kontext eines Virtuellen Produkts.
- CATIA Knowledge: CATIA Knowledge ermöglicht die wiederholenden Tätigkeiten und Geschäftsprozesse bei gleichzeitiger Beibehaltung bewährter Verfahren und Nutzung der unternehmenseigenen Wissen zu beschleunigen.
- CATIA Systems: CATIA Systems ist eine zentrale Simulationsplattform, die eine integrierte, disziplinübergreifende Simulation eines Produktes unter Berücksichtigung seiner physikalischen, elektrischen, logischen und funktionalen Eigenschaften ermöglicht.

3.2 CATIA Systems Engineering

Zur Entwicklung der komplexen mechantronischen Produkte liefert CATIA V6 eine offene, integrierte Systems Engineering Lösung, die einen RFLP-Ansatz umsetzt. Mit dieser Lösung können die zu entwickelnde Systeme nach der RFLP-Reihenfolge modelliert werden. Und das logische Modell und physikalische Modell des Systems können auch simuliert werden.

Zur Modellierung und Simulation des Systems integriert CATIA viele standardisierte Werkzeuge z.B. Dymola für die Modellierung und Simulation mechatronischer Systeme und Controlbuild oder AUTOSAR builder für die Darstellung und Simulation des Embedded Systems. Das PLM-System für CATIA dient zur Verwaltung der Entwicklungsprozess, wobei die Beziehungen, die zwischen den vielen verschiedenen Systemartefakte existieren, die diese komplexen Produkte definieren, definiert und navigiert werden können. Dadurch wird eine vollständige Rückverfolgbarkeit zwischen der Systemspezifikation und Systemimplementierung, -verifikation und -validation gewährleistet. Der gesamte disziplinübergreifende

Entwicklungsprozess kann durch CATIA SE verwaltet und unterstützt werden, von der Modellierung über der Simulation bis hin zur Verifikation und Management.[99]

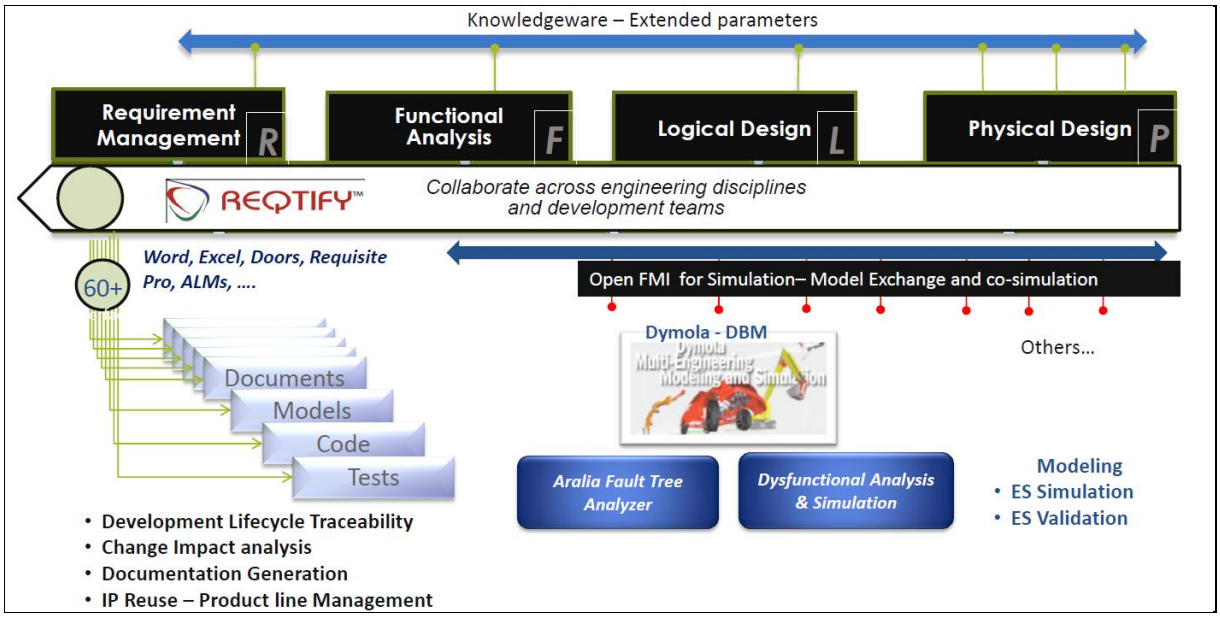

Abbildung 19: Open Systems Engineering Solution [100]

3.2.1 V6 PLM Backbone

PLM-System dient dazu, den gesamten Lebenszyklus eines Produkts von der Entwicklung bis hin zum Recycling zu begleiten, zu managen und womöglich zu steuern. Es ist sinnvoll die Systems Engineering Lösung basierend auf dem PLM-System aufzubauen. Dazu liegt der Schwerpunkt des PLM-Systems darauf, den gesamten Produktentstehungsprozess, bzw. die Entwicklung und Fertigung eines Produkts zu managen. Für SE hat PLM System die folgenden Aufgaben:[101]

- die an komplexen Produktsystemen beteiligten Disziplinen müssen während der Produktentstehung synchronisiert und aufeinander abgestimmt werden.
- schafft die Möglichkeit, die Systeme auf ihre multidisziplinären Funktionen zu simulieren, zu testen und zu validieren.
- sowie die Möglichkeit, alle während der Entwicklung, Produktionsplanung und Fertigung entstehenden digitalen Modelle und Daten in ihrem Zusammenhang zu verwalten.

---

[99] Vgl. http://www.claytex.com/products/catia-systems/ (15.05 2013)

[100] Edward A.Ladzinski:Collaborative Systems Engineering for the Industrial Equipment Industry, 2011. http://www.3ds.com/fileadmin/COMPANY/EVENTS/DSCC_2011/PDF/NOV8_435PM_IE_Ladzinski_V3.pdf (18.05.2013)

[101] Vgl. http://www.plmportal.org/was-hat-systems-engineering-mit-plm-zu-tun.html (18.05.2013)

ENOVIA V6 ist die PLM-Lösung von CATIA V6. Es bietet eine vernetzte Onlineumgebung für alle Aktivitäten, die im Zusammenhang mit dem Produktlebenszyklus stehen. Dies ermöglicht eine bereichsübergreifende Zusammenarbeit, bei der alle am Produktlebenszyklus Beteiligen parallel mit derselben Datenversion arbeiten können. [102] Mit ENOVIA V6 können alle Entwicklungsingenieure, die aus verschiedenen Disziplinen sind, unabhängig von ihren Standorten disziplinübergreifend über den gesamten Entwicklungsprozess zusammenarbeiten, von der abstrakten bis zur konkreten Produktdefinition. Die R, F, L, P- Definitionen eines Produkts werden dadurch zusammengeführt.

ENOVIA V6 verwaltet die während des Produktentstehungsprozess entstehenden Informationen, die die Informationen aus verschiedenen CAD-Anwendungen und auch die produktbeschreibenden Informationen sämtlicher Disziplinen: Konstruktion, Simulation, Fertigung, Dokumentation etc. umfassen, in einer zentralen PLM Datenbank.[103] Diese zentrale PLM Datenbank liegt für alle Anwendungen zugrunde und stellt jeder Person, die am Produktentstehungsprozess beteiligt ist, die notwendigen und aktuellen Informationen zur Verfügung. Die Anwender können unabhängig vom Zeitpunkt und Standort durch das Internet mit den Zugriffsrechten auf die Datenbank direkt zugreifen, um die von ihnen gewünschten Informationen auszulesen und zu bearbeiten. Da die Informationen den Anwendern in einer einheitlichen, durchgängigen Oberfläche angezeigt werden, ist es nicht nötig zwischen verschiedenen Anwendungen zu wechseln.[104]

ENOVIA V6 ermöglicht das Lösen der obigen PLM-Aufgaben für Systems Engineering. Es stellt das Backbone für Systems Engineering. Durch Integration von ENOVIA V6 liefert CATIA V6 demzufolge eine kollaborative PLM-Systems Engineering-Lösung.

3.2.2 Requirement Engineering

Dassault Systèmes bietet eine Requirement Engineering-Lösung für den ersten Schritt des RFLP-Ansatzes. Sie erlaubt den Entwicklern die „Stimme des Kundens“ zu erfassen und übersetzt die externen Anforderungen auf interne Anforderungen für die neuen Produkte und Systeme.

---

[102] Vgl. https://www.cenit.de/de/plm/lsungenprodukte/enovia/enovia-v6.html (19.05.2013)

[103] Vgl. https://www.cenit.de/de/plm/lsungenprodukte/enovia/enovia-v6/dokumentenmanagement.html (19.05.2013)

[104] Vgl. https://www.cenit.de/de/plm/lsungenprodukte/enovia/enovia-v6.html (19.05.2013)

Requirement Engineering besteht aus zwei Kernelemente: Anforderungsmanagement und Anforderungsrückverfolgbarkeit. Anforderungsmanagement ist ein Prozess der Identifikation, Dokumentation, Kommunikation, Verfolgung und Verwaltung von Anforderungen sowie Kontrolle der Änderungen.[105] Anforderungsverfolgbarkeit ist die Fähigkeit zur Verfolgung des Lebenszyklus einer Anforderung in Vorwärts- und Rückwärts-Richtung, von ihrem Ursprung über ihrer Entwicklung und Spezifikation bis hin zu ihrer Einführung und Nutzung.[106] Um diese zwei Aufgaben auszuführen, werden zwei Werkzeuge genutzt: ENOVIA Requirement Central für Anforderungsmanagement und Reqtify für Rückverfolgbarkeit und Impact-Analyse.[107]

ENOVIA Requirement Central unterstützt den Anforderungsmanagementprozess, wobei die Kundenanforderungen erfasst und auf Design- und Produktanforderungen übergesetzt werden, die das neue Produkt definieren. Mit ENOVIA Requirement Central können die Entwicklungsteams die Konsistenz bei der Erfassung der Kunden- und Marktanforderungen sowie Regulationen in einer verteilten Umgebung gewährleisten. Es erlaubt die Anforderungsingenieure, die Anforderungen aus Microsoft Word und Excel zu erfassen und zu übertragen. Anforderungsingenieure definieren und strukturieren die Kapiteln und Anforderungen in Microsoft Word-Dokumenten. Durch Hervorhebung und Markierung der Kapiteln und Anforderungen können sie erfasst und dann in die ENOVIA-Datenbank importiert werden. Bei dem Import ist es möglich die Struktur der Anforderungen zu erhalten. Eine Spezifikation wird inklusive des Anforderungsinhalts und –Strukturs erzeugt. Die erfassten Daten umfassen Rich-Text-Format, Tabellen, Aufzählungszeichen, Bilder, Symbole und 3D-XML- Informationen. Aus Microsoft Excel können die Anforderungen mit von Benutzer konfigurierten Formaten übergetragen werden.[108] Das durch ENOVIA V6 dargestellte Anforderungsmodell kann über die bidirektionale Schnittstelle mit RFLP-Modell von CATIA V6 synchronisiert werden.[109]

---

[105] Vgl. http://en.wikipedia.org/wiki/Requirements_management (20.05.2013)

[106] Vgl. Niekamp, 2009, S10

[107] http://www.claytex.com/products/catia-systems/requirements-engineering/ (20.05.2013)

[108] Vgl. http://www.3ds.com/products/enovia/products/enovia-v6/v6-portfolio/d/collaborative-innovation/s/governance-user/p/requirements-central/?cHash=f957afda68bb8007a08945ded16af2e8 (25.05.2013)

[109] Vgl. http://www.em.ag/themenloesungen/systems-engineering (25.05.2013)

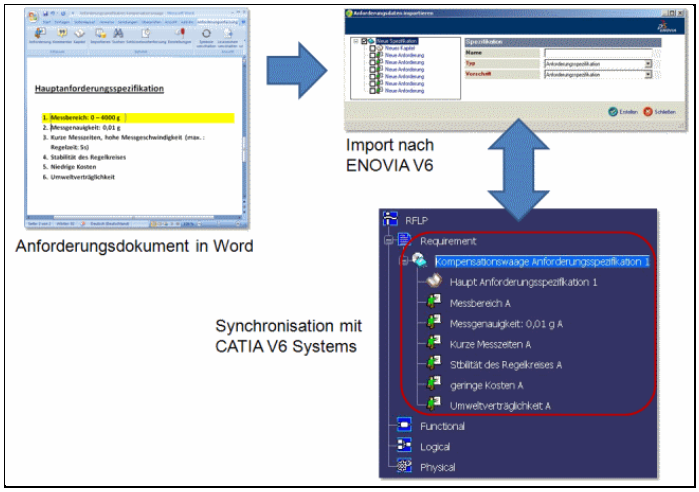

Abbildung 20: Anforderungsmodell in CATIA V6 [110]

Reqtify ist ein Requirement Engineering-Werkzeug zur Verwaltung der Nachverfolgbarkeit und Impact-Analyse über den gesamten Software- und Hardwareentwicklungszyklus. Im Gegensatz zu anderen Requirement traceability-Werkzeugen arbeitet Reqtify nicht mit einer Datenbank, sondern auf den Anforderung-beschriebenen Dokumenten. Durch Dateien, ihre Verknüfungen zueinander und ihre Tracebility-Elementen wird das zu analysierende Projekt definiert.[111] Dann kann das Projekt analysiert werden, um die Inkonsistenzen wie unerfüllte Anforderungen und die Auswirkungen der Änderungen aufzudecken.

Reqitfy bietet über 60 Konnektoren zur Verbindung mit den verschiedenen Systementwicklungswerkzeugen wie Word, UML/SysML-Werkzeug, Design-Werkzeuge etc. Dies ermöglicht es die Dateien aus verschiedenen Quellen zu erfassen, und die Beziehungen zwischen den verschiedenen Datenquellen festzuhalten. Die Anforderungen können dadurch mit dem Spezifikations-, Implementierungs-, und Überprüfungsprozess verknüpft werden.[112] (siehe Abbildung 19)

3.2.3 Functional Architecture

Requirement Engineering erfasst die Kundenbedürfnisse und erzeugt eine Anforderungsstruktur, die einen Kontext bietet, um zu bestimmen, wofür wir das

---

[110] http://www.em.ag/themenloesungen/systems-engineering (25.05.2013)

[111] Marco Pohl, Christian Scheel: Entwicklung verteilter eingebetteter Systeme, 2004. http://swt.cs.tu-berlin.de/lehre/eks/ws0304/SlidesPaper/ausarbeitung%20reqtify.pdf (26.05.2013)

[112] http://www.claytex.com/products/reqtify/interfaces/ (26.05.2013)

Produkt entwickeln. Folgend wird eine funktionale Systemstruktur darstellt, die einen Einblick in Bezug darauf bietet, welche Bedingungen das Endprodukt erfüllen muss, um die gegebenen Ziele zu erreichen.

Eine funktionale Systemarchitektur definiert die Struktur und Verhalten eines Systems. Ein System wird strukturell dargestellt, einschließlich funktionaler oder struktureller Beziehungen zwischen den einzelnen Komponenten oder Subsystemen hinsichtlich der technischen Anforderungen.[113] Mit der funktionalen Systemstruktur können die verschiedenen Funktionen im Gesamtsystem identifiziert, und die Anforderungen zeitgerecht eingehalten werden.

CATIA System Architecture Design ermöglicht die Modellierung der funktionalen Architektur und logischen Gliederung eines Systems. Es bietet die Funktionen zuerst zur Aufteilung eines Systems, um die verschiedenen Funktionen dieses System zu identifizieren, die den gegebenen Anforderungen entsprechen. Dann können die Systemarchitekten ein logisches Modell definieren, das eine technologische Lösung zum Erfüllen dieser Funktionen bestimmt. Die Anforderungsstruktur kann mit der Funktionsstruktur und der logischen Struktur verknüpft und in einem 2D-Graph in Form von logischen Bausteinen mit „Haupt-und Unterfunktionen“ visuell dargestellt werden.

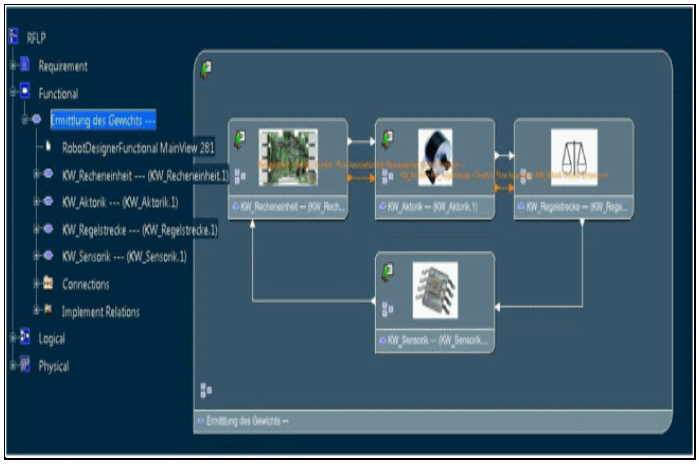

Abbildung 21: funktionales Modell in CATIA V6 [114]

3.2.4 Logical Design

Nachdem die Funktionsstruktur eines Systems dargestellt ist, wird dann die logische Struktur in Bezug auf die Funktionsstruktur dargestellt. In dieser werden die logischen

[113] Vgl. http://www.3ds.com/de/products/catia/solutions/catia-systems-architecture/ (28.05.2013)
[114] http://www.em.ag/produkte/catia-systems/ (28.05.2013)

Komponenten verknüpft, die die in Funktionsstruktur dargestellten Funktionen erfüllen müssen. Sie beschreibt das logische und physikalische Verhalten und ist damit das Lösungskonzept zur Darstellung der geometrischen Modelle.

In CATIA V6 wird das Simulationswerkzeug Dymola integriert, um die logischen Modelle zu modellieren und zu simulieren. Dymola basiert auf der objektorientierten Modellierungssprache Modelica, welche besonders zur Modellierung der komplexen mechatronischern Systeme geeignet ist. Mit der können die interdisziplinären Modelle nicht nur komfortabel definiert, sondern auch mit Hilfe einer Simulationsumgebung effizient simuliert werden. [115] Medolica Libraries enthält zahlreiche Modellkomponenten aus verschieden Disziplinen, die zur Darstellung der interdisziplinären Modelle gedacht sind. Im Modelica-Modell werden die Komponenten und Schnittstellen mathematisch mit Differential, algebraische und diskrete Gleichungen beschrieben. Ein Modelica-Modell ist somit ein Gesamtgleichungssystem und kann mittels eines Lösungsalgorithmus gelöst werden. Deshalb sind die in Modelica formulierten physikalischen Modelle gut für Simulationen geeignet.

Dymola ist die Modellierung- und Simulationsumgebung von Modelica. Die Dymola-Umgebung ist völlig offen, so dass Benutzer die neue Komponenten leicht einführen oder die vorhandenen Komponenten modifizieren können um die eigenen individuellen Anforderungen zu erfüllen.[116] Somit ermöglicht Dymola integrierte und gleichzeitige Modellierung und Simulation der Komponenten und Subsysteme aus verschiedenen technischen Disziplinen einschließlich der Embedded-Steuerung und einer Vielzahl von physikalischen Domänen sowie der komplexen Wechselwirkungen zwischen ihnen. Mit der Hilfe vom Functional Mock-up Interface (FMI) kann Dymola auch mit anderen Simulationswerkzeugen interagieren (z.B. MATLAB und Simulink).

Die Systemmodellierung und -simulation ermöglicht den Konstrukteuren das dynamische Verhalten eines Systems zu berücksichtigen und zu analysieren. Das funktionale Verhalten des Systems wird grafisch simuliert (auch Functional Mock-up genannt). Dadurch kann übergeprüft werden, ob das modellierte System die Anforderungen erfüllt. Eine frühzeitige Validation wird ausgeführt. Unerwartete Interaktionen zwischen den Komponenten oder Subsysteme können aufgedeckt werden, damit die Lösungskonzepte optimiert werden können. Diese Simulation

[115] Vgl. M.Otter, C.Schweiger. Modellierung mechatronischer Systeme mit Modelica. http://elib.dlr.de/12172/1/otter2004-vdi-modelica.pdf (04.06.2013)

[116] Vgl. Vasaiely, 2009, S15

unterstützt ebenso die Konstrukteure bei der Entscheidung, welche Alternative das beste Lösungskonzept darstellt.

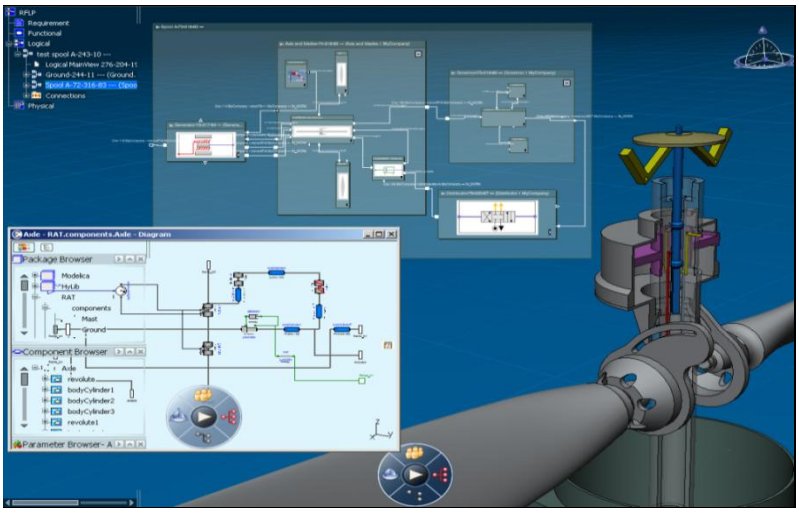

Abbildung 22: logisches Modell in CATIA V6 [117]

3.2.5 Physical Design

Im letzten Schritt soll das physikalische Design bzw. das geometrischen 3D-Modell des zu entwickelnden Systems konstruiert werden. Von diesem virtuellen Produktmodell können die üblichen technischen Zeichnungen für Fertigung abgeleitet werden. Wichtig ist, dass im Sinne des Digital Mock-Up (DMU) das 3D- Modell als ein möglichst wirklichkeitsgetreues Versuchsmodell verwendet wird, um die teuren, realen Produktprüfungen durch Computersimulationen zu ersetzen.

Da die mechatronischen Produkte nicht nur mechanische Komponenten enthalten, sondern auch die elektronische und elektrische Komponenten sowie Embededd System, brauchen die Entwicklungsingenieure eine 3D-Modellierungslösung, die verschiedene Arte von 3D-Modelle darstellen kann. CATIA Engineering bietet eine Multi-CAD-Lösung zur Unterstützung verschieder Entwicklungsprozesse. z.B. Mechanical System, Electrical Design, Electronic Engineering, Fluid Systems Design und Embedded Systems usw. Um die Komplexität des multidisziplinären Designs zu bewältigen wird CATIA V6 durch die sogenannte Collaborative Mechatronic Solution unterstützt. Diese Lösung liefert eine kollaborative Umgebung um die Werkzeuge zu integrieren und die Darstellung und Simulation der disziplinübergreifenden Designs effizient auszuführen. Projektdaten einschließlich der mechanischen Daten,

[117] http://www.plmportal.org/wo-steht-dassault-systemes.html (06.07.2013)

elektrischen Daten und Softwaredaten werden in einer einheitlichen Produktdefinition erfasst. Das ermöglicht den Datenaustausch zwischen den verschiedenen Werkzeugen und bricht damit die kulturellen Barrieren zwischen den verschiedenen Ingenieurdisziplinen.[118] Dadurch werden die Entwicklungsprozesse verschiedener Disziplinen synchronisiert und ihre Ergebnisse im Produktdesign effektiv miteinbezogen.

Für die Simulation ist es sinnvoll die Modelle aus der Mechanik mit solche aus Elektronik und Software zu koppeln und gemeinsam zu simulieren. Mit Hilfe der standardisierten Schnittstelle FMI können verschiedenen Simulationswerkzeuge integriert werden, damit eine kombinierte Simulation möglich ist. Das Verhaltensmodell und das 3D-Modell können gemeinsam simuliert werden. So ist es möglich die Wechselwirkung der Disziplinen in der jeweilig anderen zu berücksichtigen.

3.3 Zusammenfassung

CATIA V6 ist ein kollaboratives Entwicklungswerkzeug. Mit CATIA V6 kann das mechatronische System nach dem RFLP-Ansatz von Requirement Engineering über die Functional Architecture, die Logical Design bis hin zum Physical Design entwickelt werden.

Die Anforderungen können mit Hilfe von ENOVIA Requirement Central erfasst und gemanagt werden. Es ermöglicht die Übertragung der Anforderungen von Word oder Excel in die ENOVIA-Datenbank. Ein Anforderungsmodell wird in der Anforderungsphase dargestellt. Dann wird das Funktionsmodell mit Hilfe von CATIA System Architecture Design dargestellt, wobei die Gesamtfunktion des zu entwickelnden Systems definiert und in Teilfunktionen zerlegt bzw. konkretisiert wird, die die in Anforderungsmodell definierten funktionalen Anforderungen erfüllen müssen. Zur Darstellung des logischen Modells integriert CATIA das Modellierungswerkzeug Dymola, welche die Modellierungssprache Modelica verwendet. Mit dieser Modellierungssprache kann das disziplinübergreifende System in einem integrierten Modell modelliert werden. Und auch eine frühzeitige Simulation des funktionalen Verhaltens ist möglich. Zum Schluss sind die 3D-Modelle

[118] Vgl. Collaborative Mechatronic Solution http://www.3ds.com/fileadmin/Industries/High-Tech/Pdf/solution-briefs/collaborative-mechatronics-engineering-high-tech-solution-brief.pdf (10.07.2013)

darzustellen und zu simulieren. CATIA liefert mehrere Engineering-Werkzeuge zur Entwicklung verschiedener Engineering-Disziplinen. Und die Collaborative Mechatronic Solution lässt die Disziplinen effizient kombinieren und zusammenarbeiten. Ein gemeinsames Produktmodell kann erstellt und für alle Disziplinen genutzt werden. Zur Unterstützung des Entwicklungsprozesses wird eine PLM-Lösung eingesetzt, mit der der RFLP-Ansatz ohne Schnittstelle und Medienbrüche durchgängig ausgeführt werden kann. Alle vier Bereiche greifen auf eine Zentraldatenbank zu. Die in verschiedenen Phasen erstellten Modelle können somit beliebig verknüpft werden.

SysML ist eine Modellierungssprache zur Modellierung der mechatronischen Systeme für MBSE. CATIA V6 liefert einen RFLP-Ansatz zur Entwicklung der mechatronischen Systeme. Wenn ein System mit SysML modelliert ist und das dadurch erstellte Systemmodell in eine Entwicklungsumgebung wie V6-System eingesetzt wird, dann ist es von Interesse, wie das System basierend auf dieser Spezifikation mit CATIA V6 weiter entwickelt werden kann. In folgenden Kapiteln wird zuerst ein Beispielsystem mit SysML modelliert. Die dadurch erstellten Modelle sollen die Informationen enthalten, mit denen dieses System weiter entwickelt werden kann. Dann werden die Weitere Anwendung des SysML-modelle in CATIA V6 analysiert.

4 Modellierung eines Pedelecs mit SysML

4.1 Pedelec als Modellierungsbeispiel

Pedelec (Pedal Electric Cycle) ist ein spezifisches Elektrofahrrad, welches beim Treten durch einen Elektromotor unterstützt wird. Es bildet einen neuen Trend im Radverkehr ab, da der Fahrer mit weniger Köperkraft längere Stecken fahren und eine höhere durchschnittliche Geschwindigkeit halten kann. Pedelec ist ein mechatronisches Produkt. Es enthält nicht nur mechanische Komponente sondern auch die elektronische Komponente und Software-Komponente. Somit wird das Pedelec als das Modellierungsbeispiel aufgenommen.

Da die meisten funktionalen Anforderungen an den Pedelec-Antrieb gestellt werden, stellen wir den Fokus die Darstellung auf das Antriebsystem. Folgende Fragen werden durch dieses Beispiel beantwortet: Wie wird ein interdisziplinäres Systemmodell mit SysML aufgebaut, wie schauen die SysML-Diagramme aus und welche Informationen können aus SysML-Diagrammen abgelesen werden. Im Kapitel 5 wird darauf eingegangen, wie diese mit CATIA-System weiterverarbeitet werden können.

4.2 Definieren der Anforderungen mit dem Anforderungsdiagramm

Eine Anforderung spezifiziert eine Fähigkeit oder einen Zustand, der erfüllt werden muss, eine Funktion, die das System ausführen muss, oder eine Leistung, die das System erreichen muss. Die Anforderungen sollen klar und eindeutig ausgedrückt werden, und die Bedürfnisse der Stakeholders richtig und ausreichend reflektieren.

Für modellbasierte Systementwicklung ermöglicht es SysML die Anforderungen und ihre Zusammenhänge in einem Anforderungsdiagramm zu modellieren. Die Anforderungen werden mit dem UML-Element „Klasse“ dargestellt und mit dem Stereotyp <<requirement>> gekennzeichnet. Für jede Anforderung können die Eigenschaften definiert werden. Durch die Kennung (ID) lässt sich die Anforderung eindeutig identifizieren, und durch den Text wird der Inhalt der Anforderung beschrieben. Sowohl die funktionale als auch die nichtfunktionale Anforderungen können in dem Anforderungsdiagramm modelliert werden.

Die Anforderungen können untereinander oder mit anderen Modellelementen in Beziehungen stehen. Zur Modellierung dieser Zusammenhänge bietet SysML eine Reihe von Beziehungen an: <<containment>>, <<drive>>, <<satisfy>>, <<verify>>, <<refine>>, <<trace>> und <<copy>>. Die Containment-Beziehungen werden verwendet um eine Anforderung und ihre untergliederten Anforderungen zu verbinden. Eine Drive-Beziehung entsteht zwischen einer Anforderung und einer von ihr abgeleiteten Anforderung. Wenn eine Anforderung durch eine Systemfunktion oder eine Systemkomponente erfüllt wird, benutzt man die Satisfy-Beziehung. Die Testfälle, die zur Überprüfung der Anforderungen dienen, können auch in SysML-Modellen definiert werden. Zwischen den Testfällen und den Anforderungen gibt es die Verify-Beziehung. Die Refine-Beziehung wird genutzt, um anzuzeigen, dass ein Modellelement eine Verfeinerung einer textuellen Anforderung. Normalweise entsteht diese Beziehung zwischen einem Anwendungsfall und einer Anforderung. Falls die Beziehung allerdings nicht ausdrücken soll, dass eine Anforderung durch ein anderes Modellelement realisiert wird, sondern lediglich ein nicht weiter definierter Zusammenhang zwischen der Anforderung und dem Modellelement besteht, dann wird der Trace-Beziehung verwendet. Und die Copy-Beziehung wird genutzt, um die Wiederverwendung einer Anforderung in einem anderen Ort zu unterstützen. Diese Beziehungen können genutzt werden, um die Rückverfolgbarkeit der Anforderungen sicherzustellen, die Änderungen der Anforderungen und Design zu managen und zu gewährleisten, dass die Anforderungen erfüllt und verifiziert sind.

Abbildung 24 zeigt das Anforderungsdiagramm vom Pedelec, in dem die Anforderungen von Pedelec und deren Beziehungen zueinander zusammenhängend dargestellt sind. Die Top-Level-Anforderung ist *„Pedelec Spezifikation“*. Es enthält die funktionale Anforderung wie „*Einsatz der Unterstützung*“ und nichtfunktionale Anforderung wie „*Gewicht*“. Durch die Enthältbeziehung (containment) sind sie mit der Top-Level-Anforderung verbunden. Die Anforderung „*Tretkraft messen*“ ist aus den Anforderungen „*Einsatz der Unterstützung*“ und „*Unterstützung passt der Trittkraft“* abgeleitet und kann durch das Modellelement <<Block>> „*Drehmomentsensor*“ erfüllt werden. Und die Anforderung „*Beschleunigung*“ soll mit dem Testfall „*Beschleunigungsanalyse*“ getestet werden.

Bei dem verwendeten SysML-Werkzeug können nicht nur die ID und Text für die Anforderung definiert werden, sondern auch weitere Eigenschaften: „source“- die Quelle der Anforderung, „kind“- die Type der Anforderung, „VerifyMethod“- die

Testmethode für die Anforderung, „risik“ - Risiko der Anforderung und „status“- die Zustand der Anforderung.

4.3 Beschreibung der Funktionalitäten mit dem Anwendungsfalldiagramm

Anwendungsfalldiagramm ist ein Verhaltensdiagramm. Es zeigt uns das erwartete Verhalten des zu entwickelnden Systems in einer bestimmten Sicht. Der Anwendungsfall beschreibt die Funktionalität eines Systems aus Sicht der Anwender bzw. welche Leistungen das Systems für Anwender zur Verfügung stellt. Die Anwender, externe Systeme oder Menschen, können mit dem System interagieren, und werden durch Akteure beschrieben.

Sowohl der Akteur als auch der Anwendungsfall können durch Generalisierung klassifiziert werden. Ein Anwendungsfall kann einen oder mehrere Anwendungsfälle enthalten und durch die Include-Beziehungen mit denen verbindet werden. Ein Anwendungsfall kann auch durch andere Anwendungsfälle erweitert werden und durch die Extend-Beziehungen mit denen verbindet werden. In dem Diagramm müssen die Akteure mit den Anwendungsfällen, in denen sie beteiligt sind, verbunden werden.

Das Anwendungsfalldiagramm von „*Pedelec Bedienen*“ in Abbildung 25 beschreibt einige Funktionalitäten des Pedelecs bei der Anwendung. Der Akteur ist der Radfahrer. Das Subjekt Pedelec wird durch den Radfahrer angewendet, um die Ziele, die durch die Anwendungsfälle definiert sind, zu erreichen. Ein Hauptanwendungsfall ist „*Pedelec Fahren*“. Durch die Generalisierung kann es in zwei Subklassen „*Fahren ohne elektrischer Antriebsunterstützung*“ und „*Fahren mit elektrischer Antriebsunterstützung*“ unterteilt werden. Die beiden spezialisierten Anwendungsfälle teilen sich die gemeinsame Funktionalität vom Anwendungsfall „*Pedelec Fahren*“, aber sie haben auch eigene einzigartige Funktionen. Anwendungsfall „*Fahren mit elektrischer Antriebsunterstützung*“ enthält den Anwendungsfall „*Elektroantrieb einschalten*“, d.h. wenn der Anwendungsfall „*Fahren mit elektronischer Unterstützung*“ ausgeführt ist, ist der Anwendungsfall „*Elektroantrieb einschalten*“ ebenfalls ausgeführt. Außerdem ist der Anwendungsfall „*Fahren mit elektronische Unterstützung*“ durch die Anwendungsfällen „*Positive proportionale Unterstützung*“, „*Motor abschalten*“ und „*Unterstützungsstufen wählen*“ erweitert. Solche erweiterte Anwendungsfälle können nur unter bestimmten Bedingungen

ausgeführt werden. z.B. „*Positive proportionale Unterstützung*“ kann nur beim Treten ausgeführt wird.

Die Anforderungen stehen oft in Beziehung mit den Anwendungsfällen, weil die Anwendungsfälle die übergeordneten Funktionalitäten oder Ziele des Systems repräsentieren. Manchmal muss ein Anwendungsfall mit einer textuelle Anwendungsfallbeschreibung ergänzt werden. Die Anwendungsfallbeschreibung kann auch als SysML-Anforderung erfasst werden. In diesem Fall kann der Anwendungsfall mit der Anforderung durch die „refine“ Beziehung verbunden werden. z.B. Die Anforderung „*Positiv proportionale Unterstützung*“ und der Anwendungsfall „Positiv proportionale Unterstützung“. Die Anwendungsfalldiagramme werden somit oft in der Anforderungsanalyse eingesetzt.

4.4 Definieren des Systems und seines externen Umfelds mit dem Blockdefinitionsdiagramm

Blockdefinitionsdiagramm ist ein Strukturdiagramm, in dem die Systembausteine definiert und in einer hierarchischen Struktur verbunden werden.

Der Block (oder Systembaustein) ist die modulare Struktureinheit für Modellierung der Systemstruktur. Es kann zur Definition einer Systemkomponente, der Art des Systems, eines durch das System fließenden Gegenstands, oder einer externen oder abstrakten Einheit verwendet. Ein Block wird durch seine Eigenschaften definiert, die „Parts“, „References“ und „Values“ umfassen. „Values“ sind die quantifizierbare Eigenschaften eines Blocks wie Gewicht oder Geschwindigkeit. „Parts“ sind die Teile eines Blocks, die per Komposition verbunden sind. „References“ sind die Eigenschaften, deren Werte sich auf Teile von anderen Blocks beziehen.

Für Systemmodellierung ist es wichtig zu identifizieren, was die externen Systeme sind, die direkt oder indirekt mit dem System interagieren. Das Blockdefinitionsdiagramm von „*Pedelec Domain*“ in Abbildung 26 definiert das Pedelec und seine externen Systeme. Der Top-level Block ist „*Pedelec Domain*“. Es hat die Parts „*Fahrer*“, „*Pedelec*“ und „*Physikalische Umgebung*“ und die Kompositionsbeziehungen der Blöcke untereinander. Und der Block „*Physikalische Umgebung*“ besteht aus „*Externe Einheiten*“, „*Atmosphäre*“ und „*Straße*“. Die „*externe Einheiten*“ präsentieren alle physikalischen Objekte, die mit Fahrer interagieren können, z.B. andere Verkehrsmitteln oder Ampeln etc. Die Interaktion zwischen

„*Fahrer*“ und „*externe Einheiten*“ kann beeinflussen, wie der „*Fahrer*“ mit dem „*Pedelec*“ interagiert. z.B. wenn die Ampel rot ist, muss der Fahrer die Bremsen betätigen. Das Symbol Multiplizität „0...*“ repräsentiert eine unbestimmte Anzahl von externen Einheiten. Der Block „*Pedelec*“ kann natürlich weiter in seinen Komponenten unterteilt werden. Pedelec und seine Komponente werden in einem zusätzlichen Blockdefinitionsdiagramm dargestellt. (siehe Abbildung 32) Das Blockdefinitions-diagramm stellt eine Systemhierarchie dar, von der höchsten Ebene System-Domain (z.B. Pedelec Domain) bis hinunter zu den Blocks, die die Systemkomponenten repräsentiert (z.B. die Pedelec Komponenten).

4.5 Beschreibung des Systemkontexts mit dem internen Blockdiagramm

Parts, die einem Block gehören, können durch sogenannten Konnektoren in einem internen Blockdiagramm miteinander verbunden werden. Es beschreibt die interne Struktur des Blocks.

Der Konnektor spezifiziert eine Beziehung zwischen zwei Parts und bietet den beiden Parts die Möglichkeit, miteinander zu interagieren. Ein Part interagiert mit seiner Umgebung oft durch die Interfaces. Im SysML können die Interfaces durch die sogenannten Ports definiert werden. Der Port gehört dem Part und beschreibt einen Interaktionspunkt zwischen dem Part und seiner Umgebung. Der Konnektor kann auch zwei Ports verbinden, die zwei unterschiedlichen Parts angehören.

Eine wichtige Information, die auch im internen Blockdiagramm gezeigt werden muss, ist der Informationsobjektfluss. Es beschreibt an einem Konnektor, dass konkrete Objekte zwischen den Parts transportiert werden. Es kann z.B. Kraft, Energie, oder Information sein. Die Ports, über denen die Objekte hinein oder herausfließen können, heißen Objektflussports.

Bevor das Verhalten eines Systems detaillierter beschrieben werden kann, ist es sinnvoll den Systemkontext in einem internen Blockdiagramm darzustellen, in dem das Subjekt und die beteiligten Akteure den Modellelementen entsprechen. Der Systemkontext beschreibt das Umfeld des zu entwickelnden Systems und die Interfaces zwischen dem System und den Fremdsystemen. Das System „*Pedelec*“ und seine Fremdsysteme „*Fahrer*“ und „*Physikalische Umgebung*“ sind schon in dem Blockdefinitionsdiagramm definiert. Sie können durch die Konnektoren

im internen Blockdiagramm verbunden werden und stellen dadurch die interne Struktur vom Block „*Pedelec Domain*“ dar. Dieses interne Blockdiagramm spezifiziert den Pedelec Kontext. (siehe Abbildung 27)

Der „*Fahrer*“ hat eine Verbindung mit den „*externe Einheiten*“, da der Fahrer durch Sehen oder Hören auf die Verkehrssituation reagieren muss. Zwischen dem „*Fahrer*“ und „*Pedelec*“ existieren viele Interfaces, z.B. Linksfuß-Linkspedal-Interface, Finger-Bediengerät-Interface oder Linkshand-Hinterradbremse-Interface. Die auf den Konnektoren dargestellten schwarzen Pfeile repräsentieren die Objekte, die zwischen dem „*Fahrer*“ und dem „*Pedelec*“ fließen, z.B. die „*Trittkraft*“ fließt von Port „*Linksfuß IF*“ bis Port „*Linkspedal IF*“ oder der „*Unterstützungsstufe Auswahl*“ fließt von Port „*Finger IF*“ bis Port „*Bediengerät IF*“. Zwischen dem „*Pedelec*“ und der „*Straße*“ gibt es zwei Interfaces: Vorderrad-straße-Interface und Hinterrad-straße-Interface. Wenn ein Vorderradantrieb für das zu entwickelnde Pedelec entschieden wird, existiert die Traktion als Objekt nicht nur zwischen dem Hinterrad und Straße sondern auch zwischen dem Vorderrad und Straße.

Die „*Traktion*“ ist die Ausgabe von dem zu entwickelnde System „*Pedelec*“. So muss das Pedelec die Funktionen enthalten, durch die die Traktion erzeugt wird. Zur Beschreibung dieser Funktionen sollen folgende Verhaltensdiagramme verwendet werden: Aktivitätsdiagramm, Sequenzdiagramm, oder Zustandsdiagramm.

4.6 Modellierung des ablaufbasierten Verhaltens mit dem Aktivitätsdiagramm

Im SysML ist eine Aktivität ein Formalismus für Beschreibung des Verhaltens eines Systems, das eine Transformation von Inputs in Outputs beschreibt. Während dieser Transformation werden verschiedene Aktionen in einer Reihenfolge durchgeführt. Ein Aktivitätsdiagramm ist somit ein Verhaltensdiagramm und dient vor allem zur Modellierung ablaufbasierter Verhalten. Es ist analog zu einem typischen funktionalen Ablaufdiagramm, aber es hat erweiterte Funktionen um das Verhalten exakt zu spezifizieren, z.B. das Ausdrücken der Beziehung zu den strukturellen Aspekten des Systems (Blocks oder Parts).

Aktionen sind die Basiselemente der Aktivitäten und beschreiben, wie die Aktivitäten durchgeführt werden. Eine Aktion kann als eine Funktion angesehen werden. Sie nimmt die Eingaben auf und produziert die Ausgaben. Die Eingaben und Ausgaben

werden als Tokens genannt. Die Tokens werden auf den Eingangspuffer und den Ausgangspuffer gelegt, bis sie bereits von einer Aktion konsumiert werden. Die Eingangs- und Ausgangspuffer werden als Pins genannt. Ein Token kann z.B. eine Information oder ein physikalisches Objekt repräsentieren.

Die Aktionen müssen nacheinander durch die Aktivitätskante verbunden werden. Die Aktivitätskante kann man in Objektfluss und Kontrollfluss unterschieden werden. Der Objektfluss beschreibt, wie das Objekt zwischen den Aktionen fließt. Ein Objektfluss verbindet den Ausgabepin einer Aktion mit dem Eingabepin einer anderen Aktion um das Token durchlaufen zu können. Es kann diskret oder kontinuierlich sein. Der Kontrollfluss beschreibt dann die Bedingungen, wann und in welcher Reihenfolge die Aktionen ausgeführt werden. Ein Kontrolltoken auf einem eingehenden Kontrollfluss aktiviert eine Aktion, um die Aufführung zu starten, und ein Kontrolltoken wird angeboten, wenn eine Aktion komplett ausgeführt ist.

Außerdem bietet SysML auch die Kontrollknoten wie initial, final, join oder fork etc. Sie werden zur Kontrolle des Ablaufs der Tokens angewendet, damit die Ablaufreihenfolge der Aktionen exakt spezifiziert werden kann.

Die übergeordneten Funktionalitäten eines Systems sind durch die Anwendungsfälle in dem Anwendungsfalldiagramm definiert. Diese übergeordneten Funktionalitäten müssen weiter in Detail beschrieben werden. Dafür können die Aktivitätsdiagramme verwendet werden. In der Abbildung 28 wird ein Aktivitätsdiagramm gezeigt, welche die Aktivität „*Antriebsteuern*“ vom Anwendungsfall „*Pedelec Fahren*“ beschreibt. Der Fahrer und das Pedelec sind die Beteiligten dieses Anwendungsfalls. Sie werden demzufolge durch die Aktivitätspartitionen repräsentiert. Die in den Partitionen stehenden Aktionen spezifizieren die Funktionen, die von dem Fahrer und dem Pedelec erfüllt werden müssen.

Wenn die Aktivität aufgerufen ist, wird auf dem Startknoten ein Kontrolltoken platziert und somit der Ablauf gestartet. Dann werden die Aktionen: „*Linkspedal treten*“, „*Rechtspedal treten*“, „*Kettenblätter auswählen*“, „*Ritzel auswählen*“, „*Unterstützungsstufe auswählen*“, „Vorderrad*bremsen betätigen*“ und „*Hinterradbremsen betätigen*“ initiiert, die der Fahrer erfüllen muss. Die Ausgaben der „*Linkspedal treten*“ und „*Rechtspedal treten*“ sind die Trittkräfte, die als kontinuierliche Eingaben in der Aktion „*Mechanische Antriebskraft liefern*“ fließen. Und die Ausgaben der „*Kettenblätter auswählen*“ und „*Ritzel auswählen*“ sind die Auswahlbefehle, die als diskrete Ausgaben in der in der Aktion „*Mechanische Antriebskraft liefern*“ fließen.

Die Aktion „*Mechanische Antriebskraft liefern*“ produziert dann die Traktion um den Pedelec anzutreiben. Ihre andere Ausgabe ist „*Drehmoment*“, die zum Steuern der elektrischen Antriebskraft notwendig ist.

Eine weitere in der Partition „*Pedelec*“ stehende Aktion ist „*elektrische Antriebskraft liefern*“. Ihre Eingaben sind „*Drehmoment“*, „*Unterstützungsstufe Auswahl*“, und „*Bremsbetätigung“*. Sie produziert auch die Traktion. Wenn das System das Signal von „*Elektroantrieb abschalten*“ empfängt, beendet die Aktion „*elektrische Antriebskraft liefern*“ bei dem Endknoten. In diesem Fall kann das Pedelec nur mit mechanischer Kraft angetrieben werden.

Aktivitätsdiagramm kann in verschiedenen Detaillierungsgraden entwickelt werden. Nach dem Prinzip der schrittweisen Verfeinerung wird das Modell sukzessive konkretisiert. „*Mechanische Antriebskraft liefern*“ und „*elektrische Antriebskraft liefern*“ sind zwei Top-Level-Aktionen. Als Hauptfunktionen können sie weiter in Teilfunktionen zerlegt werden, dadurch eine Reihenfolge von Teilfunktionen dargestellt werden. Nach der Verfeinerung entsteht das Aktivitätsdiagramm von „*Antriebskraft liefern*", welche die internen Abläufe von „*Mechanische Antriebskraft liefern*“ und „*elektrische Antriebskraft liefern*“ zeigt. (siehe Abbildung 29)

4.7 Beschreibung der Zustände mit dem Zustandsdiagramm

Ein Zustandsdiagramm ist auch ein Verhaltensdiagramm. Während das Aktivitätsdiagramm das ablaufbasierte Verhalten eines Systems oder Teilsystems in Bezug auf den Objekt- oder Kontrollfluss beschreibt, beschreibt das Zustandsdiagramm das ereignisbasierte Verhalten eines Systems oder Teilsystems in Bezug auf seine Zustände und die Zustandsübergange zwischen denen.

Ein System befindet sich immer in einem Zustand, der eine Menge von Wertekombinationen repräsentiert, z.B. Werte von Eigenschaften eines Blocks. Ein Zustand hat einen Name und gegebenenfalls ein internes Verhalten, das aufgrund von definierten Ereignissen ausgeführt wird. Neben dem internen Verhalten besitzt ein Zustand auch drei spezielle Verhalten: das Eintrittsverhalten (entry behavior), das Austrittsverhalten (exit behavior), und das Zustandsverhalten (do behavior), die durch die vordefinierten Ereignissen ausgelöst werden. Das Eintrittsverhalten wird nach dem Eintritt in den Zustand sofort ausgeführt. Das Austrittsverhalten wird vor dem

Verlassen des Zustands ausgeführt. Und das Zustandsverhalten wird ausgeführt, während der Zustand aktiv ist.

Das System kann durch das auslösende Ereignis von einem Zustand in den anderen bewegen. Zwischen den beiden Zuständen existiert es einen Zustandsübergang, der in SysML durch die sogenannte Transition spezifiziert wird. Sie ist eine gerichtete Beziehung zwischen dem Quellzustand und dem Zielzustand und definiert den Auslöser (trigger) und die Bedingung (guard), die zum Zustandsübergang führen, sowie das Verhalten (effect), das während des Zustandsübergangs ausgeführt wird.

Gemeinsam mit den Zuständen und Zustandsübergänge bilden die Pseudozustände die Modelelemente eines Zustandsautomaten. Pseudozustände sind kein echte Zustände sondern Knoten und dienen zu Darstellung der komplexen Zusammenhänge zwischen den Zuständen, z.B. „initial State“, „final state“, oder „choice“.

Ein System kann auch mehrere Zustände gleichzeitig annehmen. Um diese parallele Abläufe oder parallele Zustände zu modellieren sind die sogenannten Regionen notwendig. Ein Zustandsautomat kann ein oder mehrere Regionen enthalten. Eine Region kann Zustände und Transitionen enthalten. Alle Regionen zusammen beschreiben das zustandsabhängige Verhalten dieses Zustandsautomats.

Zur Beschreibung des Verhaltens des Antriebssystems vom Pedelec sind zwei Zustandsdiagramme dargestellt. Abbildung 30 zeigt den Zustandsautomat „*Mechanisches Antriebssystem*“, und Abbildung 31 zeigt den Zustandsautomat „*elektrisches Antriebssystem*“. Das mechanische Antriebsystem befindet sich zunächst im Zustand „*Stillstand*“. Das Ereignis „*Pedale treten*“ löst eine Transition vom Zustand „*Stillstand*“ zum Zustand „*Betrieb*“ aus. Der Zustand „*Betrieb*“ hat zwei Regionen. Eine Region beschreibt den Schaltvorgang zwischen drei Kettenblätter. Die andere Region beschreibt den Schaltvorgang zwischen sechs Ritzel. Die beiden Regionen zusammen beschreiben das Verhalten des mechanischen Antriebssystems, dass 18 Gänge während des mechanischen Antreibens gewechselt werden können. Das Kreissymbol mit dem „H“ ist die Notation für die sogenannte flache Historisierung. Sie bewirkt, dass der letzte aktive Unterzustand der zughörigen Region gespeichert wird. Das Mechanische Antriebsystem war z.B. letztmals mit „*2.Gang-Kettenblatt*“ und „*3.Gang-Ritzel*“ in Betrieb. Wenn es wieder in Betrieb ist, dann wird der zuletzt aktive Zustand direkt aktiviert. Das heißt, der „*Mechanischer Antrieb*“ ist nicht mit der initialen Zustandskonfiguration „*1.Gang-Kettenblatt*“ und „*1.Gang-Ritzel*„ wieder in

Betrieb sondern mit der Konfiguration „*2.Gang-Kettenblatt*“ und „*3.Gang-Ritzel*“. Das Ereignis „*Pedale nicht treten*“ löst eine Transition vom Zustand „*Betrieb*“ zurück zum Zustand „*Stillstand*“ aus.

In einem anderen Diagramm befindet sich das elektrische Antriebssystem zunächst im Zustand „*aus*“. Das Ereignis „*Einschalten*“ löst eine Transition vom Zustand „*aus*“ in den Zustand „*ein*“ aus. Wenn das elektrische Antriebssystem im Zustand „*ein*“ ist, bleibt es zunächst im „*Bereitschaftszustand*“. Wenn das Pedelec getreten wird und die Trittfrequenz größer als Null ist, wird der Zustand „*Antreiben*“ initiiert. Bei dem „*Antreiben*“ kann die Unterstützung des elektrischen Antriebs zwischen drei verschiedenen Stufen geschaltet werden. Das Ereignis „*bremsen*“ führt zum Zustandsübergang von „*Antreiben*“ zurück zu „*Bereitschaftszustand*“. Auch wenn die Geschwindigkeit des Pedelecs größer als 25 km/h ist, muss der Zustand des elektrischen Antriebssystems wieder in „*Bereitschaftszustand*“ zurückgehen.

Während der Zustand „*Betrieb*“ des mechanischen Antriebssystems aktiv ist, wird ein Zustandsverhalten „*mechanische Antriebskraft liefern*“ ausgeführt. Und während der Zustand „*Antreiben*“ des elektrischen Antriebssystems aktiv ist, wird ein Zustandsverhalten „elektrische Antriebskraft liefern“ ausgeführt. Die beide Verhalten sind im obigen Aktivitätsdiagramm modelliert.

4.8 Darstellung der Systemarchitektur mit dem Blockdefinitionsdiagramm

Nachdem das Verhalten des Systems beschrieben wurde, wir der Fokus auf die Systemarchitektur geändert. In der Systemarchitektur sollen die Systemkomponenten definiert werden, die vordefinierte Systemanforderungen erfüllen müssen. Zur Darstellung der Systemarchitektur soll das Blockdefinitionsdiagramm verwendet werden, mit dem das System hierarchisch strukturiert werden kann. Abbildung 32 zeigt die Pedelec-Architektur, in der das Pedelec in Komponenten zerlegt wurde.

Das Pedelec-System besteht aus fünf Teilsysteme: „*Elektroantrieb*“, „*Mechanischer Antrieb*“, „*Bremsanlage*“, „*Lenkung*“ und „*Körper*“. Die Teilsysteme können weiter dekomponiert werden, z.B. „*Elektroantrieb*“ ist in „*Elektromotor*“, „*Stromversorgung*“, „*Motor-Controller*“, „*Bediengeräte*“, „*Drehmomentsensor*“, und „*Bremsabschalter*“ untergeteilt. Die Anzahl von „*Bremsabschalter*“ ist 2, da es Vorderradbremse und Hinterradbremse gibt. Dabei können die

Hardwarekomponenten durch <<hardware>> gekennzeichnet werden, und die Softwarekomponenten durch <<software>> gekennzeichnet werden. In diesem Diagramm sind alle in den Blocks definierten Komponenten mit <<hardware>> kenngezeichnet. Aber die „*Controllersoftware*“ <<software>> kann durch die Zuteilungsbeziehung der Hardwarekomponente „*Motor-Controller*“ zugeordnet werden, da „*Motor-Controller*“ eine Ausführungsplattform für „*Controllersoftware*“ ist.

Das Blockdefinitionsdiagramm stellt die Beziehungen und Abhängigkeiten zwischen den Komponenten aus der Black-Box-Sicht dar, ohne deren interne Funktionsweisen detailliert zu beschreiben. Die interne Struktur eines Systems lässt sich durch das interne Blockdiagramm darstellen.

4.9 Zuweisen von Funktionen zu Komponenten durch Anwendung der Aktivitätspartition

Um zu beschreiben, wie die Komponenten eines Systems mit ihren Verhalten aufeinander wirken, werden die Aktionen einer Aktivität den Komponenten des Systems zugewiesen. Dafür kann das Konzept der sogenannten Aktivitätspartition vom Aktivitätsdiagramm verwendet werden. Jede Partition steht für einen Block und enthält die entsprechenden Aktionen, die durch diesen Block ausgeführt werden kann. Sie beschreibt eine Zuteilungsbeziehung zwischen dem Block und seinen Aktionen.

Abbildung 33 zeigt das Aktivitätsdiagramm „*Antriebskraft liefern*“ mit den Partitionen. In diesem Diagramm ist für jede Komponente der Teilsysteme „*Elektroantrieb*“ und „*Mechanischer Antrieb*“ eine Partition dargestellt. Jede Komponente ist einer oder mehreren Aktionen zugeordnet, die zur Aktivität „*Antriebskraft liefern*“ gehören. z.B. die Aktion „*Drehmoment erzeugen*“ kann durch die Komponente „*Elektromotor*“ ausgeführt werden. Diese Aktion wird deshalb in der Partition „*Elektromotor*“ gestellt. Durch diese Kombination mit Komponenten kann dieses Aktivitätsdiagramm „*Antriebskraft liefern*“ uns zusätzliche Information darüber liefern, wie die Komponenten des Pedelecs die Ausgabe „*Traktion*“ erzeugen.

4.10 Darstellung der inneren Struktur mit dem internen Blockdiagramm

Während das Blockdefinitionsdiagramm die Struktur eines Systems aus der Black-Box-Sicht beschreibt, beschreibt das interne Blockdiagramm die Struktur aus der White-box-Sicht. In dem Blockdefinitionsdiagramm werden die Komponenten eines Systems definiert, ohne dass die innere Funktionsweise der Komponenten beschrieben wird. Die Komponenten müssen miteinander verbunden werden, damit die Funktionalität des Systems durch ihr Zusammenwirken erfüllt werden können. Dieses Zusammenwirken der Komponenten wird im internen Blockdiagramm dargestellt.

Abbildung 34 zeigt das interne Blockdiagramm vom „*Antriebssystem*“. Dieses Diagramm stellt die interne Struktur vom Antriebsystem des Pedelecs dar und zeigt uns, wie die Komponenten des Antriebsystems zusammenwirken. Es enthält die Komponenten von Subsystemen „*Elektroantrieb*“ und *„Mechanischer Antrieb*“, die für das Erzeugen der Antriebskraft notwendig sind. Die Komponenten „*Vorderradbremse*“ und „*Hinterradbremse*“ vom Subsystem „*Bremsanlage*“, die bei der Betätigung den Elektroantrieb abschalten, und die Komponenten „Vorderrad“ und „Hinterrad“ vom Subsystem „Köper“, die die vom Antriebssystem erzeugte Drehmoment annehmen, werden im Diagramm dargestellt.

Der Rahmen vom Diagramm repräsentiert die Black-Box vom „*Pedelec Antriebssystem*“. Die Eingaben wie „*Tretkraft*“ oder „*Unterstützungsstufe Auswahlen*“, die von fremden Systemen kommen, sind im obigen Kontextdiagramm definiert. Aber die Eingabe „Bremsbetätigungskraft“ kommt aus der anderen internen Struktur „*Pedelec Bremsen*“. Mit diesem Objektfluss können diese zwei Diagramme in Beziehung stehen. Im Diagramm sind die Komponente über ihren Ports verbunden. Zwischen den Komponenten werden Informationen oder Objekte transportiert. Diese Informationsobjektflüsse sind nicht auf den Konnektoren dargestellt. Sie sind entsprechend den Objektflusskanten in dem Aktivitätsdiagramm „*Antriebskraft liefern*“ und können durch Zuteilungsbeziehung mit denen verbunden werden. Die Ausgaben der Black-Box sind Traktionen, die durch das Zusammenwirken erzeugt werden.

Mit dem Blockdefinitionsdiagramm “ *Pedelec System*““ zusammen bildet dieses interne Blockdiagramm eine mögliche Lösung für den Antrieb des Pedelecs. In der

Praxis sollen die Entwickler normalerweise mehrere alternative Lösungskonzepte entwickeln. Dann können sie durch die sogenannte Trade-off Analyse bewertet werden, um die beste Lösung auszuwählen. Zur Unterstützung der Trade-off-Analyse verwendet man das Zusicherungsdiagramm, das auch zu dem Strukturdiagramm gehört.

4.11 Beschreibung der Einschränkungen und Analyse der Systemleistung mit dem Zusicherungsdiagramm

Zusicherungsdiagramm, auch Parameterdiagramm genannt ist ein spezifisches internes Blockdiagramm. Es beschreibt die Einschränkungen auf die Eigenschaften eines Systems, welche anschließend durch ein geeignetes Analysewerkzeug beurteilt werden müssen. Diese Einschränkungen werden in Form von Gleichungen ausgedrückt, deren Parameter mit den Eigenschaften des Systems gebunden werden. Parameterdiagramm bildet damit die Zusammenhänge zwischen den Eigenschaften eines Systems. Jedes Parameterdiagramm erfasst eine bestimmte technische Analyse von einem Design. Mehrere technische Analysen können dann in Parameterdiagrammen erfasst werden, die mit den alternativen Konstruktionslösungen in Beziehungen stehen und dann ausgeführt werden um die trade-off-Analyse zu unterstützen.

SysML bietet die Zusicherungsbausteine (constraint block) zur Unterstützung der Darstellung des Parameterdiagramms an. Ein Zusicherungsbaustein definiert eine Gleichung und ihre notwendigen Parametern. Es kann kontextfrei sein und somit wiederverwendet werden. Blockdefinitionsdiagramm wird hier verwendet um die Zusicherungsbausteine in ähnlicher Weise wie Systembausteine zu definieren.

Die Anwendung der Zusicherungsbausteine auf das Modell findet im Parameterdiagramm statt. Die Parameter eines Zusicherungsbausteins können als seine Ports angesehen und über Bindungskonnektoren mit Parametern anderer Zusicherungsbausteine oder Eigenschaften des Systems verbunden werden. Dadurch wird ein Gleichungssystem dargestellt, das die Eigenschaften von Systembausteinen einschränkt.

Eine wichtige Performance des Pedelecs ist die Beschleunigung. Viele Eigenschaften der Komponenten vom Pedelec beeinflussen die Beschleunigungsleistung. Diese Eigenschaften der Komponenten können als Parameter in den Gleichungen für

Berechnung der Beschleunigung des Pedelecs miteinander in Beziehung stehen. Um diese parametrischen Beziehungen zwischen den Eigenschaften darzustellen und damit die Beschleunigung des Pedelecs zu analysieren, können wir das Parameterdiagramm verwenden.

Vor der Darstellung des Parameterdiagramms müssen wir zuerst die Einschränkungen (Gleichungen) und Parameter, die für die Berechnung und Analyse der Beschleunigungsleistung erforderlich sind, mit Hilfe von Zusicherungsbausteinen definieren. Im Blockdefinitionsdiagramm „*Pedelec Beschleunigung*“ (Abbildung 35) steht der Block „*Pedelec Beschleunigungsanalyse*“ als Kopfzeile in der obersten Ebene. Die Zusicherungsbausteine sind als Bestandteile durch die Kompositionsbeziehungen mit dem verbunden. Die Zusicherungsbausteine beschreiben die Gleichungen für „*Rollwiderstand*“, „*Luftwiderstand*“, „*Mechanische Antriebskraft*“, „*Elektrische Antriebskraft*“, „*Gesamtkraft*“, „*Beschleunigung*“ sowie „*Integration der Beschleunigung*“. Als Beispiel beschreibt das Zusicherungsbaustein „*Beschleunigung*“ die Grundgesetze der Bewegung, wobei die Beschleunigung „α“ gleich der Kraft „f“ durch die Summe vom Fahrradgewicht „m_R“ und Fahrergewicht „m_F“ ist. Die Gleichung ist in der Constraint-Abteilung definiert und die Parameter in der Parameters-Abteilung.

Der Block „*Pedelec Beschleunigungsanalyse*“ referenziert auch auf den Block „*Pedelec Domain*“, welche die Subjekte von der Beschleunigungsanalyse darstellt. Die Parts des „*Pedelec Domain*“ sind das zu entwickelnden System „*Pedelec*“, seine Benutzer „*Fahrer*“ und die mit Pedelec interagierte Außenwelt „*Physikalische Umgebung*“. Ihre Eigenschaften sollen explizit mit den Parametern der Gleichungen gebunden werden. Wie die Eigenschaften mit den Parametern gebunden werden, ist es in dem Parameterdiagramm „*Pedelec Beschleunigungsanalyse*“ dargestellt.

Das Parameterdiagramm „*Pedelec Beschleunigungsanalyse*“ in Abbildung 36 zeigt ein Netzwerk von Einschränkungen oder Gleichungen, die in dem Blockdefinitionsdiagramm „*Pedelec Beschleunigung*“ definiert sind. Der Rahmen des Diagramms repräsentiert den Block „*Pedelec Beschleunigungsanalyse*“ aus dem Blockdefinitionsdiagramm „*Pedelec Beschleunigung*“. Die Einschränkungen werden als Rechtecke mit abgerundeten Ecken gezeichnet. Die Parameter werden als kleine Quadrate notiert, die von innen an das Rechteck der angewandten Einschränkung klebt. Ein Parameter einer Gleichung kann über Bindungskonnektoren mit einem Parameter anderer Gleichungen gebunden werden. z.B. Der Parameter „F_{MA}“ in der Gleichung „*Mechanische Antriebskraft*“ ist mit dem Parameter „F_{A1}“ in der Gleichung

„*Gesamtkraft*“ gebunden. Das bedeutet, dass „F_{MA}“ in der Gleichung „*Mechanische Antriebskraft*“ gleich „F_{A1}“ in der Gleichung „*Gesamtkraft*“ ist. Die Parameter können auch mit den Eigenschaften der Systembausteine gebunden werden. z.B. der Parameter „R_{Rad}“ in der Gleichung „*Mechanische Antriebskraft*“ ist mit der Eigenschaft „*Radius*“ in dem Block „*Hinterrad*“ gebunden. Der Wert von „R_{Rad}“ ist gleich dem Wert von „*Radius*“ des Hinterrads. „*Hinterrad*“ ist ein Part vom Block „*Mechanischer Antrieb*“, das ein Part vom Block „*Pedelec*“ ist. „*Pedelec*“ gehört dem Block „*Pedelec Domain*“. Ein anderer Parameter „F_t“ in der Gleichung „*Mechanische Antriebskraft*“ ist mit der Eigenschaft „*Trittkraft*“ im Block „*Fahrer*“ gebunden, der ein Fremdsystem ist. Das beweist, dass das Fremdsystem auch über die parametrischen Beziehungen mit dem zu entwickelnden System verbunden werden kann.

Die Bindungen zwischen den Parameter und den Eigenschaften der Systembausteine zeigen uns, welche Eigenschaften der Komponenten die Leistung des Systems beeinflussen. Und die Einschränkungen zeigen uns, wie die Eigenschaften die Leistung beeinflussen. Mit Hilfe von geeigneten technischen Analysetools können die Einschränkungen des Parameterdiagramms ausgeführt und damit die Ergebnisse für die Analyse geliefert werden. Durch die Analyse kann beurteilt werden, ob die Werte der Eigenschaften eine gute Leistung bringen oder die geforderte Leistung erfüllen können. Das Pedelec soll z.B. mit 100 N Trittkraft von 0 bis 25 km/h unter 10 sec beschleunigen können. Durch die Ausführung der Einschränkungen von „*Pedelec Beschleunigungsanalyse*“ könnte ein Geschwindigkeit-Zeit-Diagramm dargestellt werden. Aus diesem Diagramm kann man ablesen, wie lange das Pedelec mit einer bestimmten Trittkraft und einer bestimmten Unterstützungsstufe sowie eines bestimmten Gangs von 0 bis 25 km/h beschleunigt. Dann können wir beurteilen, ob diese Leistung die gewünschte Leistung ist. Wenn nicht, müssen wir dann andere Konfigurationen der Komponenten überlegen.

In dem Block „*Motor*“ ist Motorleistung eine spezifische Eigenschaft, da der Motor einen bestimmten Prozentsatz der vom Fahrer erbrachten Leistung liefern muss. Das heißt, dass die Motorleistung mit der Fahrerleistung in einem bestimmten Verhältnis stehen muss. Für die Motorleistung gibt es auch eine Einschränkung, die den Wert der Motorleistung beschränkt. Diese Einschränkung können wir in einem anderen selbständigen Zusicherungsdiagramm darstellen.

Zusätzlich zu der Beschleunigungsanalyse kann man auch die Zusicherungsdiagramme zur Analyse anderer Pedelec-Leistungen darstellen, z.B. für Bremsweg, Reichweite, oder Kosten. Diese Parametrik ermöglicht die kritischen

Eigenschaften der Systembausteine zu identifizieren und die Parameter in die Analysemodelle zu integrieren.

4.12 Zusammenfassung

Im Sinne des MBSE sollen die Anforderungen, das Verhalten und die Struktur eines Systems (drei Bausteine von SE) modellhaft beschrieben werden. Die drei Bausteine stehen miteinander in Zusammenhang und dienen zusammen als Systemspezifikation für weitere Simulationen und Tests. Für das Pedelec wurden die SE-Bausteine durch Anwendung verschiedener SysML-Diagrammen modelliert. Diese Diagramme hängen natürlich miteinander zusammen und bilden die Pedelec -Spezifikation.

Die Anforderungen für das Pedelec sind in einem Anforderungsdiagramm erfasst. Das Anforderungsdiagramm enthält nicht nur funktionale sondern auch nichtfunktionale Anforderungen und dient wie die Anforderungsliste als Grundlage für die weitere Entwicklung. Um die funktionalen Anforderungen formal zu beschreiben, verwendet man die Verhaltensdiagramme. Zunächst sind die zu erwartenden Funktionalitäten vom Pedelec in Form von Anwendungsfällen in einem Anwendungsfalldiagramm dargestellt. Die Anwendungsfälle verfeinern die Anforderungen im Anforderungsdiagramm aus der Sicht des Anwenders. Die Lücke zwischen Anforderungen und Anwendungsfällen können durch die „refine“-Beziehung geschlossen werden.

Mit dem Anwendungsfalldiagramm erhält man nur eine Grobübersicht der übergeordneten Funktionalitäten des Pedelecs. Um diese Funktionalitäten näher zu beschreiben, wurden das Aktivitätsdiagramm und das Zustandsdiagramm verwendet. Ein Kontextdiagramm wird vor der Darstellung des Aktivitätsdiagramms und des Zustandsdiagramms erstellt. In diesem Diagramm sind die Interfaces zwischen dem Pedelec und seinen Fremdsystemen sowie die Eingaben und Ausgaben des Pedelec -Systems definiert. Dann wurde ein Aktivitätsdiagramm dargestellt, welche den gewollten Ablauf des Anwendungsfalls „Pedelec *fahren*“ beschreibt. In diesem Diagramm wurden die Aktivitäten „*Mechanische Antriebskraft liefern*“ und „*elektrische Antriebskraft liefern*“ schrittweisen verfeinert. Dadurch ergab sich eine Reihenfolge von Aktionen, die in einem weiteren Diagramm dargestellt ist. Dieses Aktivitätsdiagramm stellt den Ablauf des „*Antriebskraft liefern*“ dar und beschreibt, wie das Pedelec die Antriebskraft liefert, bzw. wie das Pedelec die Eingaben des Pedelec

-Systems zu Ausgaben transformiert. Aus einer anderen Sicht stellt sich die Frage, welche Zustände das Antriebssystem hat sowie durch welche Ereignisse die Zustände sich verändern. Um die Zustände und deren Zustandsübergänge zu modellieren, wurde das Zustandsdiagramm verwendet. Da das Antriebssystem des Pedelecs mechanischen Antrieb und elektrischen Antrieb enthält, sind zwei Zustandsdiagramme dargestellt. Das Aktivitätsdiagramm und das Zustandsdiagramm beschreiben das Verhalten des Systems aus zwei verschiedenen Sichten.

Zur Modellierung der Struktur des Pedelecs wurden das Blockdefinitionsdiagramm, das interne Blockdiagramm und das Zusicherungsdiagramm verwendet. Mit dem Blockdefinitionsdiagramm ist das Pedelec in einer hierarchischen Struktur dargestellt, wobei die Systembausteine oder Komponenten des Pedelecs definiert sind. Diese Komponenten erfüllen die Aktionen, die in vorigem Aktivitätsdiagramm dargestellt sind. Die Aktionen wurden durch die Partition mit entsprechenden Komponenten in einem Aktivitätsdiagramm dargestellt. Wenn eine Anforderung durch eine Aktion erfüllt wird, erfüllt ihre entsprechende Komponente ebenfalls diese Anforderung. Diese Aktion oder Komponente kann mit ihrer zu erfüllende Anforderung über eine <<Satisfy>>-Beziehung verbunden werden.

Die im Blockdefinitionsdiagramm definierten Komponenten vom Antriebsystem wurden dann miteinander über ihre Ports in einem internen Blockdiagramm verbunden. Dadurch entstand die interne Struktur vom Antriebsystem des Pedelecs. Dieses Diagramm zeigt, wie die Komponenten des Antriebssystems zusammenwirken um seine Funktionalitäten zu erfüllen, bzw. um die Antriebskraft zu erzeugen.

Zuletzt ist ein Zusicherungsdiagramm für die Analyse der Pedelec Beschleunigung dargestellt. In dem sind einige Eigenschaften der Komponenten und des Pedelecs sowie der Fremdsysteme über die vordefinierten Parameter der Beschränkungen verbunden, die den Gleichungen für Berechnung der Beschleunigung entsprechen. Dieses Diagramm zeigt, wie die Komponenten des Pedelecs über den parametrischen Beziehungen verknüpft sind. Dadurch wissen wir, welche Eigenschaften von Komponenten die Beschleunigungsleistung beeinflussen und wie. Durch Festlegung der Werte der Eigenschaften kann man die Beschleunigungsleistung auswerten und erkennen ob diese Konfiguration eine gute oder die erwartete Leistung bringen kann. Wenn die erwartete Leistung als eine Anforderung im Anforderungsdiagramm spezifiziert würde, kann diese Anforderung mit diesem Zusicherungsdiagramm über eine <<Verify>>-Beziehung verbunden

werden. In Praxis werden häufig mehrere alternative Lösungskonzepte für das zu entwickelnde System entwickelt. Die Zusicherungsdiagramme unterstützen somit auch die Trade-off-Analyse.

Pedelec ist ein mechatronisches System. Es enthält mechanische Funktionselemente wie mechanischer Antrieb, elektronische Funktionselemente wie elektrischer Antrieb und informationstechnische Funktionselemente wie Controllersoftware. Mit SysML kann das ganze System inklusiv allen Disziplinen modellhaft spezifiziert werden. Die oben dargestellten Diagramme spezifizieren das Antriebssystem des Pedelecs. In folgenden Entwicklungsphasen können die Mechaniker die Informationen vom mechanischen Antrieb aus den Diagrammen ablesen, und die Elektroniker können die Informationen vom Elektroantrieb aus den Diagrammen ablesen. Aber die vorhandene Spezifikation für die weitere Entwicklung des Motor-Controllers ist noch zu abstrakt. Die Funktionen sowie die interne Komponentenstruktur bzw. der elektronische Schaltplan müssen dargestellt werden. Hier muss man aufpassen und überlegen, ob die Darstellung der elektronische Schaltplan mit SysML sinnvoll ist, weil es dafür besser geeignetes und spezialisiertes Werkzeug gibt.

Diese Beispiele stellen nur einen Teil vom Systemmodell des Pedelecs dar und sind somit nicht vollständig. Um das Systemmodell zu vervollständigen muss man alle anderen Funktionalitäten modellieren und vorhandene Diagramme auf einen notwendigen Abstraktionsgrad konkretisieren. Jedes Diagramm stellt einen Aspekt des Pedelec-Systems dar. Durch Verknüpfung aller Diagramme ergibt sich das Systemmodell, welche eine Gesamtsicht des Systems in der Spezifikationsphase bildet. Das Systemmodell steht als Systemspezifikation im Mittelpunkt des MBSE um die Aktivitäten jeder Entwicklungsphase zu unterstützen.

5 Weitere Anwendung der SysML-Modelle in CATIA V6

SysML wird als ein entwicklungsbegleitendes Werkzeug während der laufenden Systementwicklung angesehen. Es wird nicht als Ersatz für bestehende Entwicklungswerkzeuge angesehen, sondern als eine sinnvolle Ergänzung.[119] Das mit SysML dargestellte Systemmodell ist das primäre Artefakt von MBSE und ein integriertes Framework für andere Modelle und Artefakte. Es bringt die Textanforderungen und das Design in Zusammenhang, liefert die erforderlichen Designinformationen zur Unterstützung der Systemanalyse, dient als der Spezifikation für Darstellung des Produktdesign und bietet den Testfällen und den davon notwendigen Informationen zur Unterstützung der Verifikation. Die Änderungen von Systemanforderungen oder –design werden zuerst an dem Systemmodell vorgenommen, und anschließend durch die Betrachtung, Verknüpfung und Artefakte in verschiedenen Disziplinen verbreitet, die von den Änderungen beeinflusst werden. Deshalb bildet SysML eine gemeinsam genutzte Basis und ein gemeinsames Fundament für verschiedene Entwicklungsdisziplinen ab.[120]

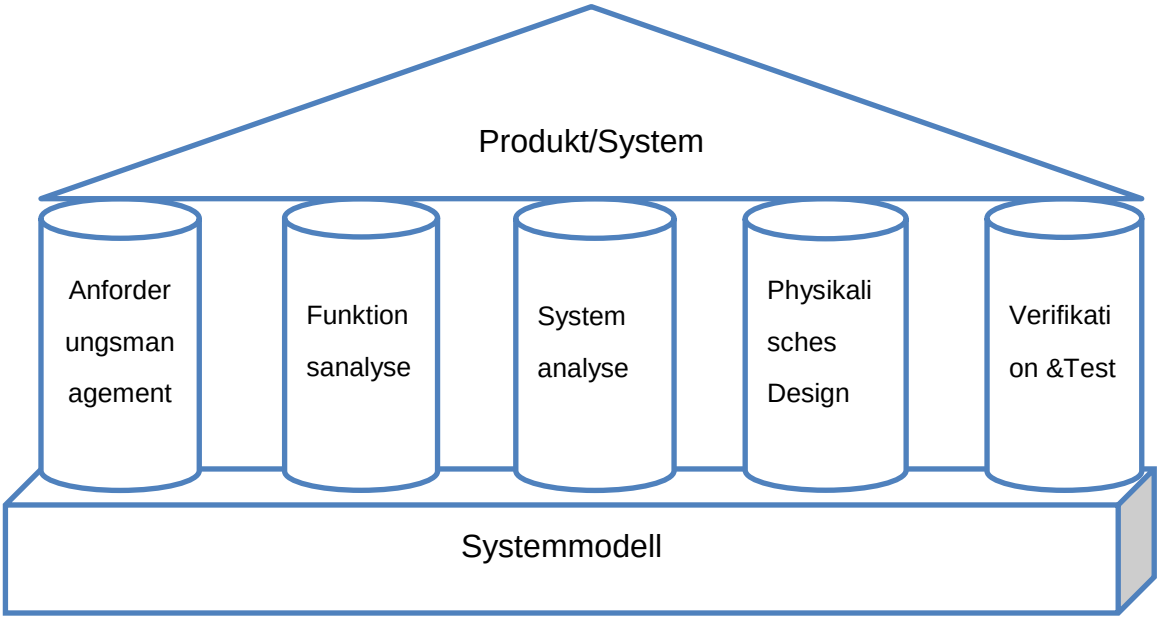

Abbildung 23: SysML bildet die Grundlage für Systementwicklung

Abbildung 23 zeigt einen Pavillon, wobei das Systemmodell, das die Anforderungen, das Verhalten und die Architektur des zu entwickelnden Systems beschreibt und

[119] Vgl. Alt, 2012, S.30
[120] Vgl. Alt, 2012, S.30

miteinander in Beziehung setzt, die Grundlage für die einzelnen Disziplinen darstellt. Anforderungsmanagement, Funktionsanalyse, Systemanalyse, Design, und Test sind einige Entwicklungsdisziplinen und stellen die Säulen dar, die auf der Grundlage des Systemmodells stehen. Das System oder das Produkt wird als Dach des Pavillons durch die Unterstützung verschiedener Disziplinen aufgebaut. Das Systemmodell wird als Spezifikation genutzt, und Details werden ab einem gewissen Punkt mit speziellen Werkzeugen weiter entwickelt und bearbeitet, die in verschiedenen Disziplinen eingesetzt werden. CATIA V6 bietet einen RFLP-Ansatz und integriert viele Werkzeuge um eine Entwicklung durchgängig zu unterstützen. Wenn das Systemmodell eines zu entwickelnden Systems bereitgestellt ist, wird analysiert, wie das System mit CATIA V6 weiter entwickelt werden kann.

ANFORDERUNGSMANAGEMENT

Diese weitere Entwicklung bedeutet, dass die Informationen aus dem Systemmodell ausgelesen und dann in anderen Werkzeugen eingeführt und weiter bearbeitet werden. Der Schwerpunkt liegt in der Übertragung der Informationen bzw. Modelldaten zwischen dem SysML-Werkzeug und anderen Werkzeugen. Das Systemmodell liefert für die Anforderungsanalyse die Anforderungsinformation, die die Anforderungen und ihre Beziehungen umfasst. Das Anforderungsmanagement-werkzeug „ENOVIA Requirement Central“ hilft CATIA bei der Verwaltung der Anforderungen. Dieses Werkzeug bietet die Schnittstellen, die die Übertragung der Textanforderungen automatisch aus Word- oder Excel-Dokument in ENOVIA-Datenbank ermöglichen. Aber es gibt noch keine Schnittstelle mit den SysML-Werkzeugen. Die Übertragung der Anforderungsinformation vom SysML-Systemmodell hin zu ENOVIA Requirement Central muss man manuell durchführen. Vor allem das Synchronisationsproblem zwischen den beiden Werkzeugen ist von Nachteil, da die Anforderungen während der Entwicklung immer aktualisiert werden. Bei jeder Veränderung der Anforderungsinformation im Anforderungsmanagement-Werkzeug muss man diese in Systemmodell manuell aktualisieren.

FUNKTIONSANALYSE

Für die Funktionsanalyse liefert Systemmodell die Verhaltensdiagramme, z.B. das Aktivitätsdiagramm „*Antriebskraft liefern*“ vom Pedelec-Systemmodell. Ein Aktivitätsdiagramm ist ähnlich wie eine traditionale Funktionsstruktur, aber es enthält noch zusätzliche Eigenschaftsinformationen um das Verhalten präzise zu

beschreiben. Man kann somit eine Funktionsstruktur in CATIA nach diesem Aktivitätsdiagramm darstellen. Diese Übertragung muss manuell durchgeführt werden. Dabei können die Erfüllungsbeziehungen zwischen Funktionen und Anforderungen auch übertragen werden, weil es in CATIA möglich ist, die Funktionen in der Funktionsstruktur mit ihren zu erfüllenden Anforderungen zu verlinken.

SYSTEMANALYSE

Das mit SysML ausgedruckte Systemmodell erfasst das System unter Aspekten von seinem Verhalten, seiner Struktur und seinen Parametern. Parameter sind die wesentlichen Eigenschaften vom SysML zur Verstärkung der Integration zwischen der Spezifikation in SysML und anderen Simulation- und Analysemodell. Systemmodellierungswerkzeug liefert die Designinformationen zu dem Simulation- oder Analysewerkzeug. Analysewerkzeug führt die Analyse durch und gibt die Analyseergebnisse bezüglich der Eigenschaftswerte wieder zum Systemmodellierungswerkzeug zurück, die im Systemmodell erfasst werden können. z.B. das Systemmodell vom Pedelec bietet die Systemstruktur, die Einschränkungen und die parametrische Beziehungen für die Beschleunigungsanalyse. Aus diesen Informationen kann ein Analysemodell abgeleitet werden. Durch die Simulation und Analyse soll eine Vorhersage von Beschleunigungsleistung bereitgestellt werden. Die Eigenschaftswerte aus diesem Leistungsergebnis können für die Systembausteine im Systemmodell zurückgeliefert werden,

Zur Simulation und Analyse des Systemverhaltens integriert CATIA V6 das Simulationswerkzeug Dymola, welche auf der Modellierungssprache Modelica basiert. Wie bereits in Kapitel 3.3.4 erwähnt, ist Modelica eine objektorientierte Modellierungssprache zur Modellierung und Simulation mechatronischer Systeme. Ein wesentlicher Aspekt von der Modelica-Modellierungsmethode ist, dass die Dynamik der Komponenten durch die deklarativen Gleichungen, und die Schnittstellen zwischen den Komponenten durch die Erhaltungsgleichung modelliert werden. Die dadurch dargestellten Modelle sind für Simulationen geeignet. SysML hat eine gute Beschreibungsfähigkeit, und Modelica hat eine gute Simulationsfähigkeit. Die nächste Frage ist, wie die Informationen zwischen SysML- und Modelica-Modelle übertragen werden können. Zurzeit gibt es eine standardisierte bidirektionale Transformation zwischen den beiden Modellierungssprache (das sogenannte SysML4Modelica). Sie wurde von OMG entwickelt und unterstützt die Implementierung des effizienten und automatischen Austausches der Informationen

zwischen SysML- und Modelica-Werkzeuge. Aber die Implementierung dieser Transformation ist noch in Entwicklung.

DESIGN

Eine wichtige Ursache für Entwicklung eines Systemmodells ist die Anforderungen und die Einschränkungen auf den Systemkomponenten zu spezifizieren, damit die Komponentenspezifikationen erstellt werden. Für Darstellung der physikalischen Systemkomponenten liefert das Systemmodell die Komponentenspezifikationen. Sie sind das primäre Output aus der Systemspezifikation. Eine Komponentenspezifikation wie eine Black-Box-Spezifikation wird in Form vom Block mit ihren Eigenschaften spezifiziert, die die Ports, Operationen und Werteigenschaften umfassen. Basierend auf diesen Komponentenspezifikationen können die Systemkomponenten mit verschiedenen spezialisierten Entwicklungswerkzeugen entwickelt werden. Die Entwicklungswerkzeuge kann vielleicht die Designverifikationsdaten zurückliefern, die demonstrieren können, dass die Designmodelle die Anforderungen erfüllen.

CATIA ist ein CAD-Werkzeug, mit dem die mechanischen Komponenten nach der Komponentenspezifikationen entworfen, implementiert und simuliert werden können. Die Embedded Softwares können mit den von Geensoft hinzugekommenen Produkten „AUTOSAR Builder“ und “Controlbuild“ entwickelt werden, die in der V6-Umgebung von CATIA integriert sind. Für die elektronischen, elektrischen und Fluid-systeme bietet CATIA Engineering eine Multi-CAD-Lösung, mit der die Leiterplattendesign, die 2D-Schaltpläne und 3D-Kabelbäume sowie die Rohr- und Schlauchleitung in Kontext eines virtuellen Produkts dargestellt werden können. Aber um die elektronischen Systeme effektiv zu entwickeln braucht man noch das spezialisierte E-CAD-Werkzeug. z.B. Pedelec-Antriebssystem enthält die mechanischen Komponenten wie Kurbelsatz, die elektronische Komponenten wie Motor-Controller und die Embedded Software wie Controllersoftware. CATIA Mechanical-Lösung ermöglicht es die geometrischen 3D-Modelle von mechanischen Komponenten des Antriebssystems darzustellen. CATIA Electronic Engineering-Lösung ermöglicht das Leiterplattdesign vom Motor-Controller im mechanischen Kontext darzustellen. Sie bietet auch eine bidirektionale Schnittstelle mit E-CAD-Werkzeugen für die Zusammenarbeit zwischen Mechanik- und Elektronik-spezialisten. Und CATIA Electrical Design-Lösung ermöglicht es den 3D-Kabelbaum des Elektroantriebs direkt auf dem mechanischen Modell abzubilden. Ein integriertes Design vom Pedelec-Antriebssystem kann damit in dieser einheitlichen Entwicklungsumgebung erfolgreich dargestellt werden. Die Controllersoftware kann

mit Controlbuild konstruiert und durch die integrierte Simulationsmöglichkeit mit dem virtuellen Produktdesign verbunden werden.

Die Komponentenspezifikationen von dem SysML-Werkzeug in den Engineering-Entwicklungswerkzeugen muss man manuell übertragen. Es gibt noch keine Schnittstelle, mit der die Information zwischen denen automatisch transformiert werden können. Die automatische Transformation ist nur noch theoretisch möglich.

VERIFIKATION & TEST

Verifikation ist ein Teil des Systementwicklungsprozess. Das entwickelte System muss überprüft werden, ob es seine Anforderungen erfüllt. Die Verifikation von V6-System Engineering erfolgt durch Simulation und Analyse. Dassault Systèmes bietet eine realistische Simulationslösung „Simulia“, die der realitätsnahen Simulation des von CATIA konstruierten physikalischen Produktdesign dient. Simulia ist auch im V6-System integriert und mit CATIA und anderen Systemsimulationswerkzeugen wie beispielweise Dymola synchronisiert.

Der Prozess der Verifikation umfasst im Allgemeinen folgende Schritte: Bestimmen der Verifikationsplan und Simulationsmethoden, Durchführen der Simulation, Analyse der Simulationsergebnisse und Erzeugen der Reports. Zur Initiierung dieses Verifikationsprozesses braucht man die Informationen von den Anforderungen und die dafür vordefinierte Testfälle. In V6-Systems sind diese Informationen im Anforderungsmanagementwerkzeug „ENOVIA Requirement Central“ gespeichert und verwaltet. SLM (Simulation Lifecycle Management) von Simulia, welches zu Verwaltung von Simulationsdaten und Prozess dient, übernimmt die Informationen aus dem ENOVIA Requirement Central und stellt dann die entsprechenden Simulationsmethoden bereit. Im SysML können die Testfälle für die Anforderungen definiert und mit denen verbunden werden. Das Systemmodell kann somit die Testfälle gleichzeitig mit den Anforderungen für die Simulation liefern. Die Übertragung der Testfälle in ENOVIA Requirement Central sowie die Anforderungen müssen manuell durchgeführt werden.

Das durch SysML dargestellte Systemmodell kann als Grundlage für den ganzen Entwicklungsprozess genutzt werden. Es kann die Spezifikationsinformationen für die Aktivitäten jeder Entwicklungsphase liefern, vom Anforderungsmanagement, über Funktionsanalyse, Systemanalyse, physikalisches Design bis hin zu der Verifikation und Test. Aber das Problem ist, dass die Informationen manuell übergetragen werden

müssen. Das verursacht den zeitlichen und personellen Aufwand und erhöht die Wahrscheinlichkeit von Übertragungsfehlern.

Für die Modellierung und Spezifikation eines Systems bietet CATIA V6 die Funktionen zur Darstellung des Anforderungsmodells und des Funktionsmodells. In diesem Funktionsmodell wird nur eine Funktionsstruktur dargestellt. SysML modelliert das System aus verschiedenen Sichten, wobei die Anforderungen, das Verhalten bezüglich der Anwendungsfälle, der Aktivitäten und der Zuständen, die Architektur und die Parameter sowie ihre Beziehungen beschrieben werden können. Im Vergleich zu CATIA V6 kann SysML das System besser beschreiben und spezifizieren. Mit SysML kann das System modelliert aber nicht simuliert werden. CATIA V6 integriert Dymola zur Simulation des Systems und dient als CAD-System vorwiegend zur Darstellung und Simulation der 3D-Modelle. Für die Anwendung von SysML-Modellen im CATIA-System liegt der Schwerpunkt bei der Systemsimulation und -analyse und der Darstellung und Simulation des physikalischen Designs. Es ist dann sinnvoll Schnittstellen zwischen dem SysML-Werkzeug, Dymola und dem CATIA CAD-Werkzeug zu realisieren, damit eine durchgängige Werkzeugkette erstellt werden kann. Die Informationen können dann zwischen denen automatisch übertragen werden und das Potenzial der modellbasierten Entwicklung kann ausgeschöpft werden.

6 Fazit

Das mechatronische System ergibt sich aus einer Vernetzung von mechanischen, elektronischen und informationstechnischen Funktionselementen. Mit diesem interdisziplinären Charakter hat die Komplexität der Entwicklung von mechatronischen Systemen stetig zugenommen. Um diese Komplexität zu beherrschen und damit mechatronische Systeme effektiv und effizient zu entwickeln zu können, wird ein interdisziplinärer Ansatz „Systems Engineering“ vorgeschlagen.

Systems Engineering integriert all diese Ingenieursdisziplinen und Fähigkeiten in einem einheitlichen Prozess, um eine ausgewogene Systemlösung als Antwort auf den Anforderungen verschiedener Stakeholders zu entwickeln. Die Bausteine des SE sind Systemanforderungen, Systemarchitektur, und systemverhalten, mit denen das mechatronische System vollständig definiert werden kann. Die dadurch definierten Systeme werden schrittweise von abstrakt bis konkret nach einer bestimmten Vorgehensweise entwickelt. Vor allem wird das V-Modell als Vorgehensmodell verwendet.

Die Anfangsphase bzw. Spezifikationsphase des Entwicklungsprozesses spielt eine Entscheidungsrolle für folgende Entwicklungsschritte. Eine vollständige, offensichtliche und disziplinübergreifende Systemspezifikation fördert die Entwicklung eines qualitativen hochwertigen Systems. Bei klassischem SE wird das mechatronische System textuell beschrieben. Solche Systemspezifikation hat eine Reihe von Schwachstellen aufgewiesen. Um alle Disziplinen in der Spezifikationsphase zu berücksichtigen und einen Zusammenhang leicht zu verstehen wird das System modellhaft beschrieben. Das MBSE wird statt dem klassischen DBSE für Entwicklung der mechatronischen Systeme eingesetzt.

Im MBSE steht ein interdisziplinäres Systemmodell im Mittelpunkt und dient als grundlegende Systemspezifikation zur Unterstützung aller beteiligten Disziplinen. Es erlaubt unterschiedliche Sichten auf die Informationen, die es beinhaltet. Über dieses interdisziplinäre Systemmodell ist es leichter die Informationen zwischen verschiedenen Disziplinen auszutauschen. Das Systemmodell entsteht dadurch, dass das System einschließend seiner Anforderungen, Verhalten und Architektur mit einer Systemmodellierungssprache interdisziplinär und modellhaft beschrieben wird. Daher ist eine beschreibungsfähige Systemmodellierungssprache notwendig. SysML ist eine international standardisierte Systemmodellierungssprache und dient der Modellierung

als ein hilfreiches Werkzeug. Für modellbasierte Entwicklung wird ein erweitertes V-Modell vorgeschlagen, welches besonders durch den RFLP-Ansatz charakterisiert wird. RFLP repräsentiert vier Phasen des Entwicklungsprozess, nach dem mechatronische Systeme vom Anforderungsmodell über Funktionsmodell, logisches Modell bis hin zum physikalischen Modell entwickelt werden können.

Die Anwendung des SE bzw. MBSE müssen in der Praxis durch Rechner unterstützt werden. Ein IT-System ist sehr nützlich, wenn es die für die Entwicklung notwendigen IT-Werkzeuge integrieren und die in Entwicklung entstehenden Daten verwalten kann. CATIA V6 und die Integration mit der PLM-Lösung ENOVIA V6 kann nun erstmals den Ansatz von SE mit verfügbaren IT-Werkzeugen unterstützen. Dazu bietet CATIA V6 einen integrierten RFLP-Ansatz, mit dem sowohl das interdisziplinäre Systemdesign als auch das interdisziplinäre Produktdesign dargestellt werden kann. Durch die Integration der Simulationswerkzeuge wie Dymola und Simulia ermöglicht es CATIA V6 auch sie integriert zu Simulieren und testen. In diesem V6-System kann das mechatronische System von Anforderung bis hin zur Verifikation und Test durchgängig und ohne Medienbrüche entwickelt werden. Somit ist CATIA V6 nicht nur eine CAD-Lösung, sondern auch eine Systems Engineering-Lösung.

Das Systemmodell hat eine Entscheidungsbedeutung für MBSE, und die Systemmodellierungssprache spielt eine wichtige Rolle von „enabler“ für MBSE. Als praktisches Beispiel ist ein Pedelec-System mit SysML modelliert. Dazu wurden die Anforderungen, das Verhalten und die Architektur sowie die Parameter eines Pedelec-Systems mit verschiedenen SysML-Diagrammen modelliert. Jedes Diagramm repräsentiert ein Aspekt des Pedelec-Systems und enthält die entsprechenden Informationen. Alle diese Diagramme verbanden sich miteinander durch bestimmte Beziehungen. Daraus ergibt sich das SysML-Systemmodell, welches ein Überblick vom Pedelec zeigt.

Dieses Beispiel hat uns bewiesen, dass ein mechatronisches System mit SysML modelliert werden kann, welches mechanische, elektronische und informationstechnische Funktionselemente beinhaltet. Dabei können folgende Vorteile von SysML erfasst werden:

- SysML ermöglicht eine funktionale Beschreibung des Systems, wobei die Komponenten des Systems, deren Beziehungen und die Funktionen der einzelnen Komponenten beschrieben werden.

- SysML beschreibt das System aus verschiedenen Sichten: die Anforderungen, die Architektur, das Verhalten und die Parametern sowie ihre Beziehungen. Das erlaubt den Entwicklern das System besser zu verstehen.
- SysML erstellt ein einheitliches und integriertes Systemmodell, über das eine effiziente Zusammenarbeit verschiedener Disziplinen in der frühen Phase realisiert wird.
- Durch die Verbindungen der Anforderungen mit anderen Modellelementen stellt SysML die Rückverfolgbarkeit sicher. Die Auswirkungen einer Veränderung können leicht erkannt werden.
- SysML beschreibt das interdisziplinäre System auf einer allgemeinen Ebene. Das erlaubt die Beteiligten verschiedener Disziplinen diese Systembeschreibung leicht zu verstehen.

Neben den Vorteilen hat SysML auch folgende Nachteile:

- Die grafischen Spezifikationen können vielleicht zu einem Missverständnis führen. Deswegen sind manchmal zusätzliche Dokumentationen notwendig um die um die grafischen Spezifikationen zu ergänzen.
- Da die SysML eine auf der UML basierende Sprache ist, sind nicht alle Systeme mit ihr leicht zu modellieren.
- Die Konstrukteure müssen qualifiziert sein, damit die Modelle effektiv und effizient dargestellt werden können.
- Die Erstellung der SysML-Modelle ist eine aufwendige Aufgabe. Für ein sehr komplexes Produkt wie ein Auto oder ein Flugzeug müssten zahlreiche Modelle dargestellt werden. Es würde einen großer Aufwand brauchen um die Modelle darzustellen, zu pflegen und konsistent zu halten.

Basierend auf einem durch SysML erstelltes Systemmodell kann das System mit CATIA V6 weiter entwickelt werden. Das Systemmodell kann jede Phase der RFLP-Ansatz von CATIA V6 unterstützen. Aber die sinnvolle Anwendung der SysML-Modelle im CATIA-System ist die Unterstützung der Systemsimulation und -analyse und die Darstellung und Simulation des physikalischen Modells. Das Problem hier ist, dass es keine Schnittstelle zwischen dem SysML-Werkzeug und SE-Werkzeugen des V6-systems gibt. Aber es gibt einige Möglichkeiten zum automatischen Austausch der Informationen zwischen demSysML-Werkzeug und anderen Werkzeugen, z.B. durch XMI (XML Metadata Interchange) oder AP233 (Application Protocol 233). Die Realisierung der Schnittstellen ermöglicht die

Erstellung einer durchgängigen Werkzeugkette, die einen weiteren Beitrag zur Steigerung der Entwicklungseffizienz und Absicherung der Qualität leisten kann.

In dieser Arbeit wurde diese weitere Anwendung der SysML-Modelle in CATIA V6 theoretisch untergesucht. Eine praktische Untersuchung fehlt noch. Als Fortsetzung dieser Arbeit kann versucht werden ein SysML- Modellsystem mit CATIA V6 praktisch weiterzuverarbeiten. Dadurch kann man diese näher analysieren.

Anhang

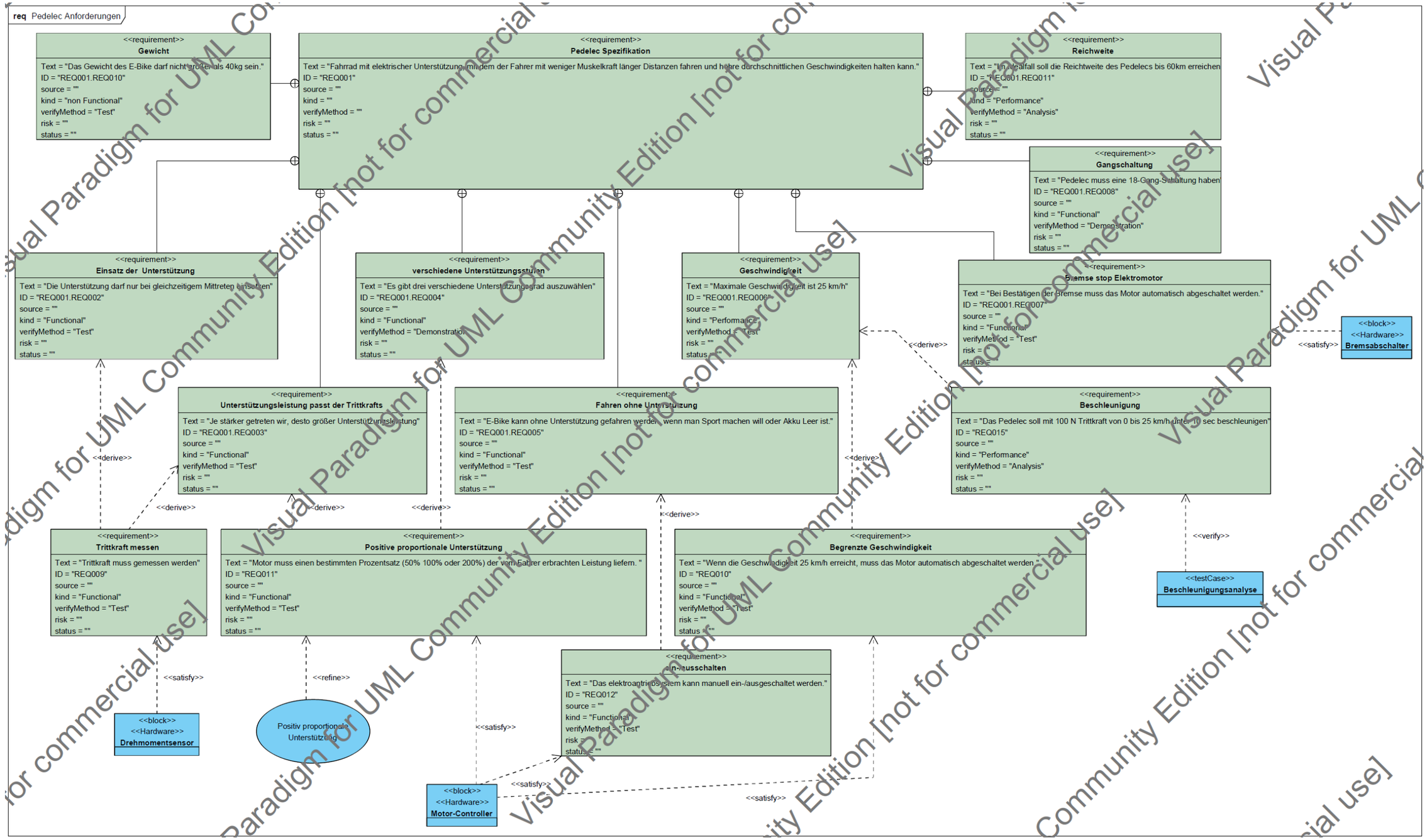

Abbildung 24: Anforderungsdiagramm vom Pedelec

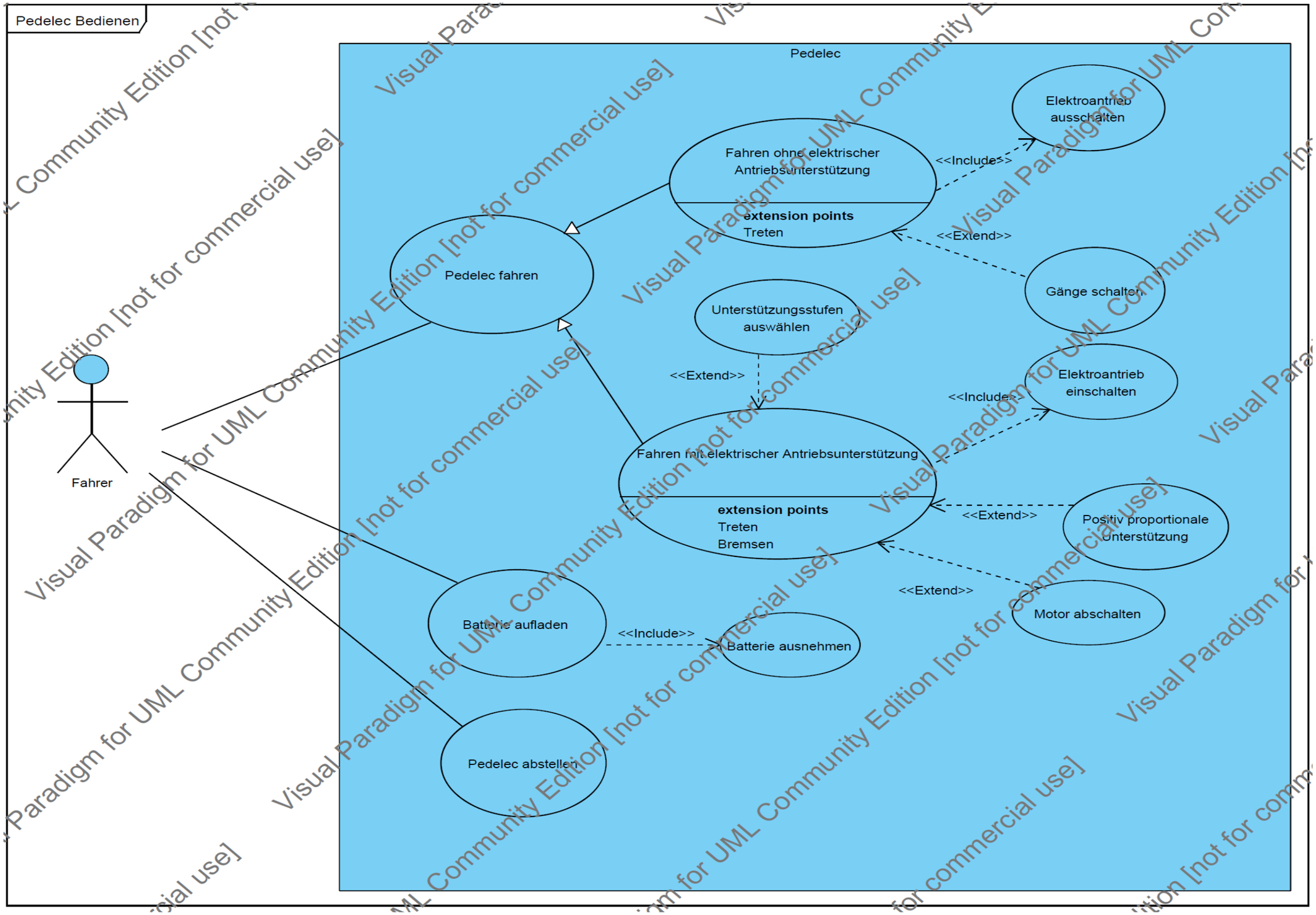

Abbildung 25: Anwendungsfalldiagramm von „Pedelec Bedienen“

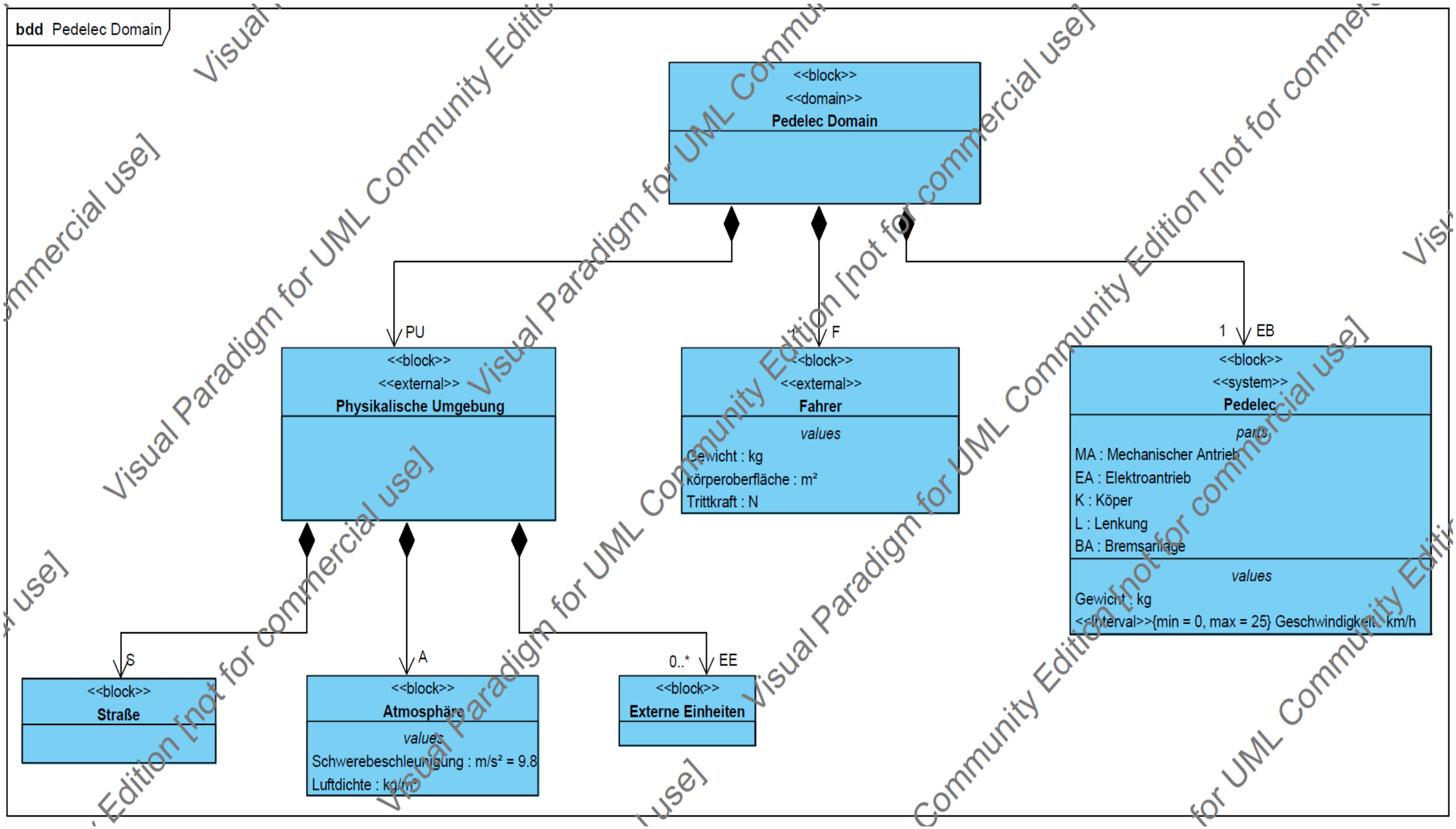

Abbildung 26: Blockdefinitionsdiagramm von „Pedelec Domain“

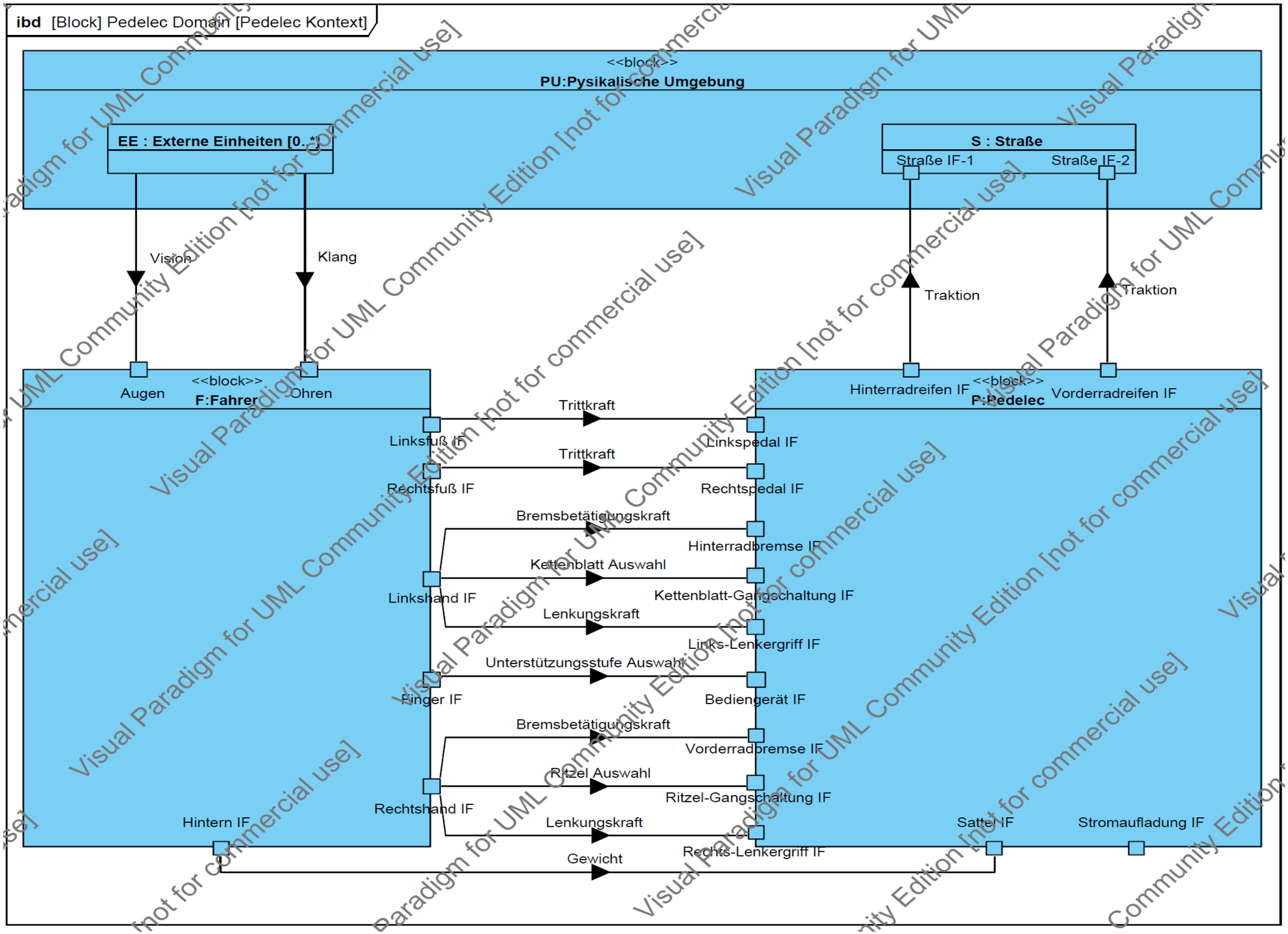

Abbildung 27: internes Blockdiagramm von „Pedelec Domain“

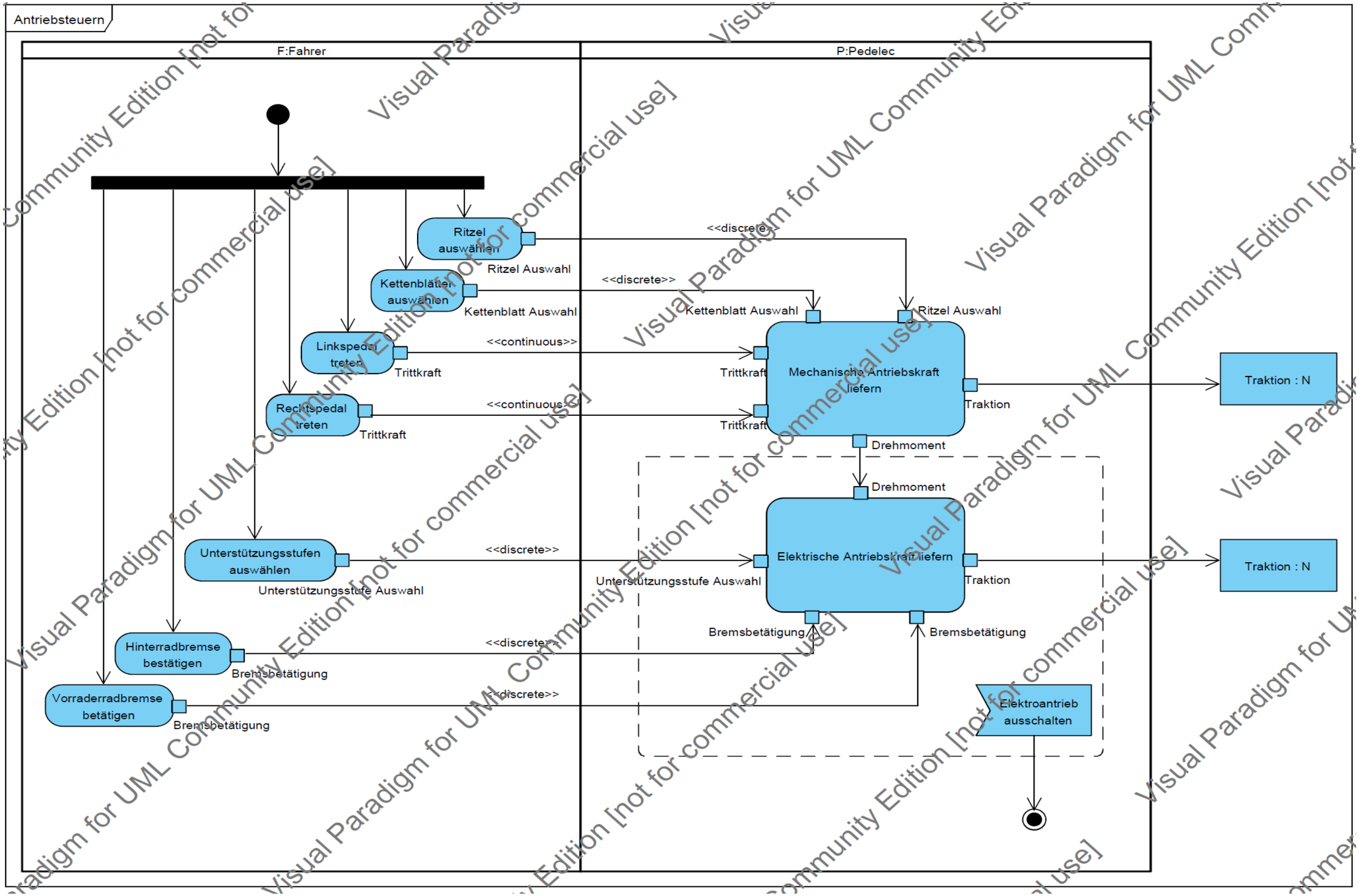

Abbildung 28: Aktivitätsdiagramm von „Antriebsteuern“

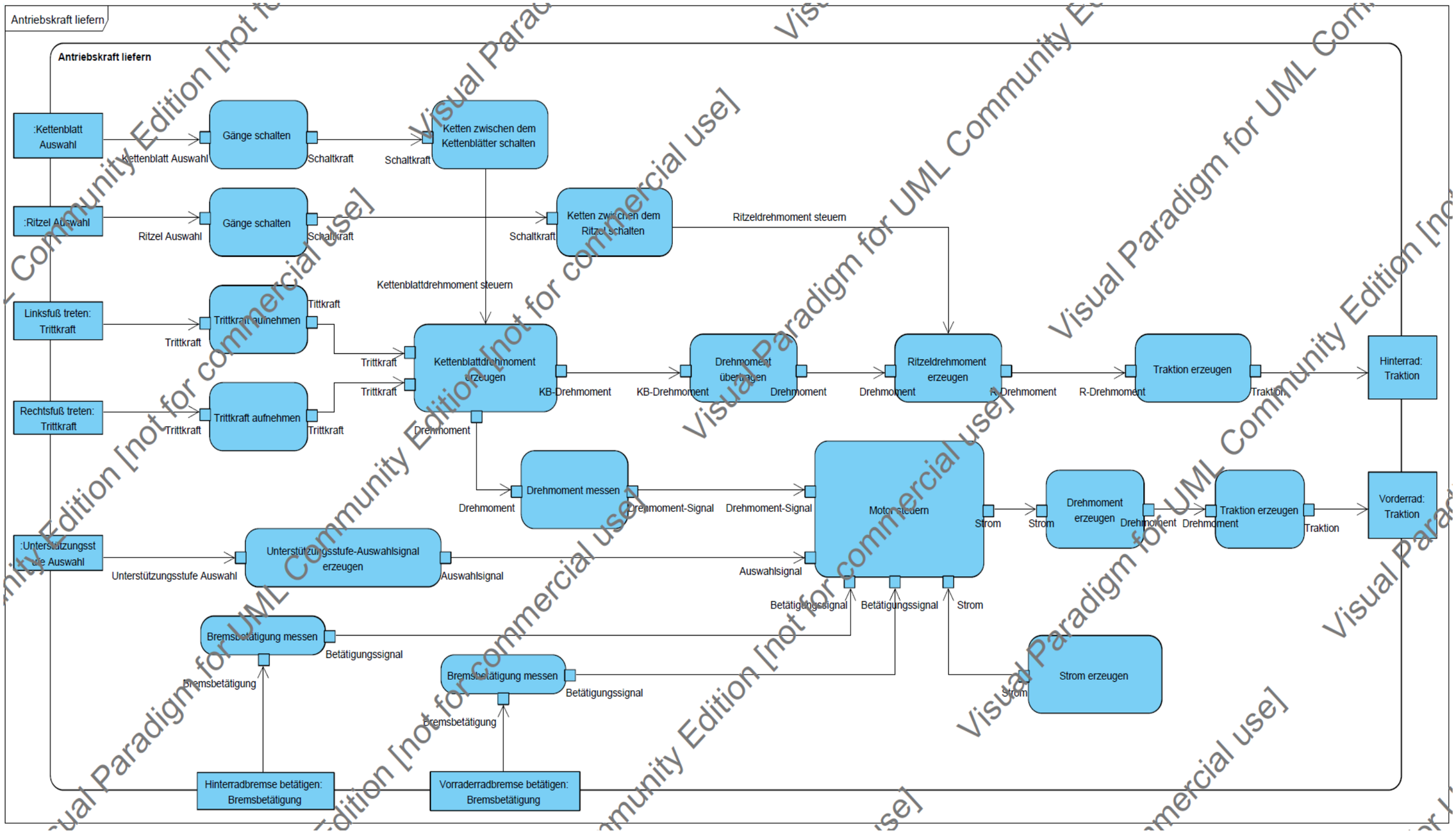

Abbildung 29: Aktivitätsdiagramm von „Antriebskraft liefern"

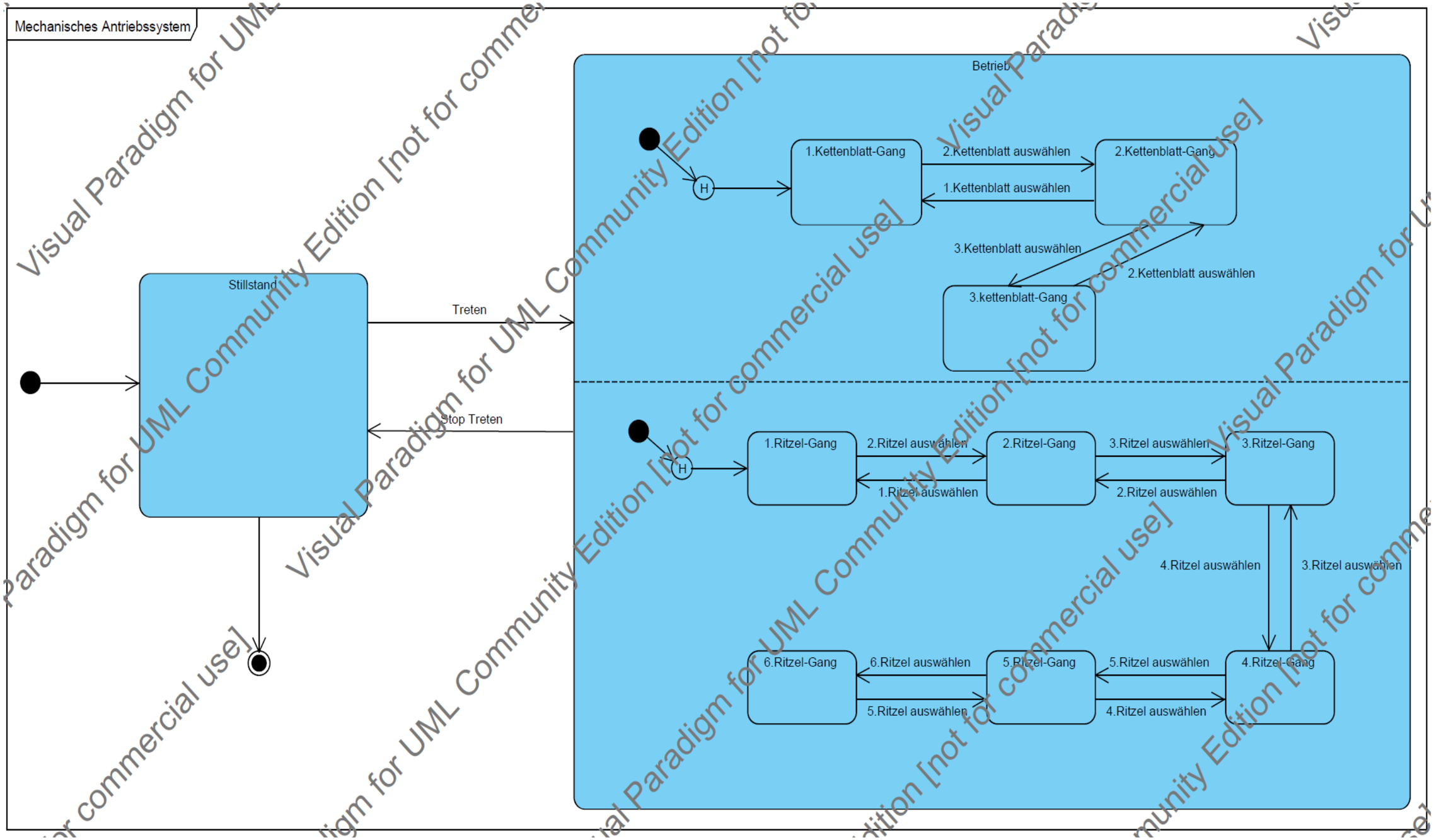

Abbildung 30: Zustandsdiagramm von „Mechanisches Antriebssystem“

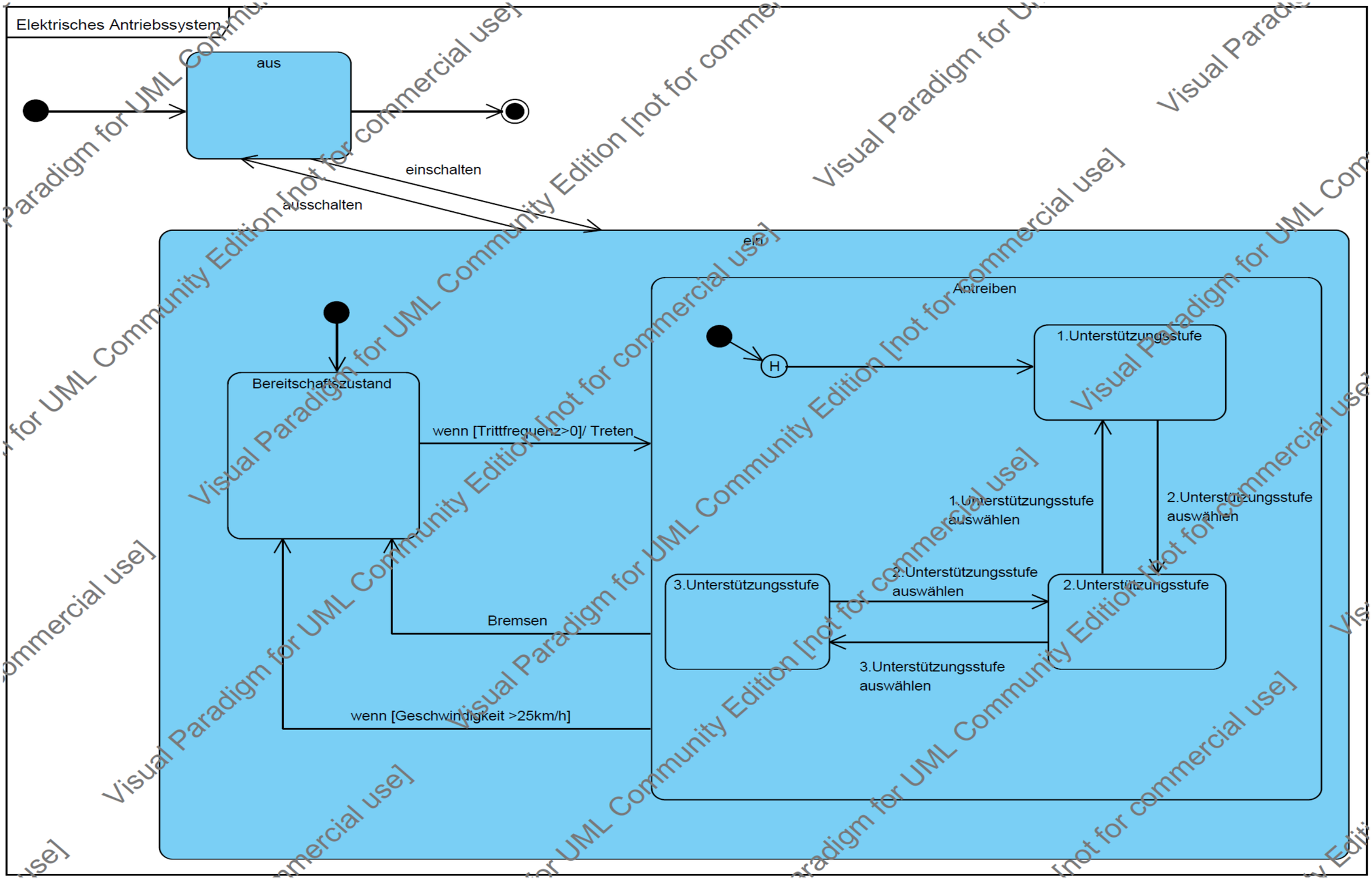

Abbildung 31: Zustandsdiagramm von „elektrisches Antriebssystem“

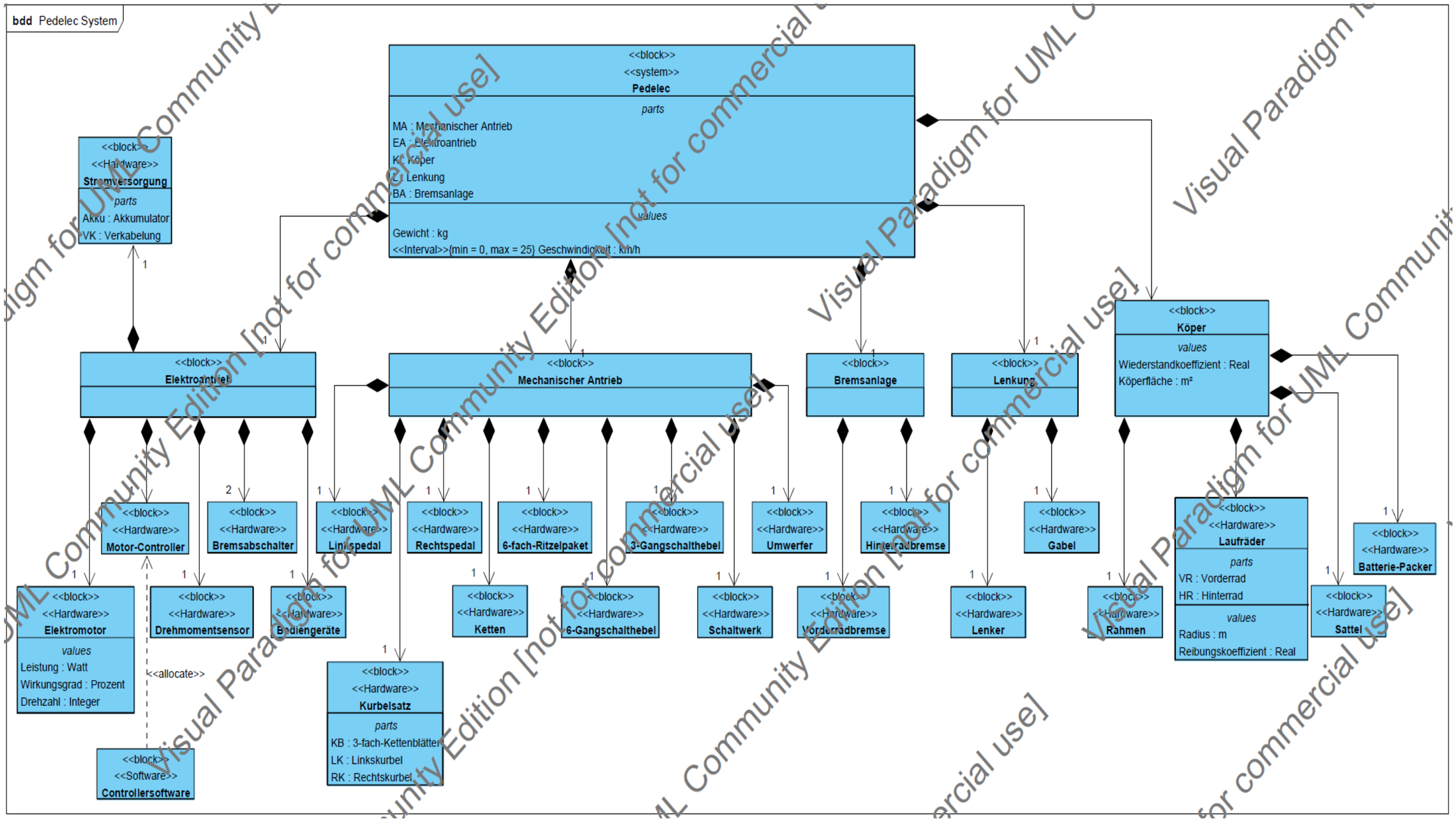

Abbildung 32: Blockdefinitionsdiagramm von „Pedelec System“

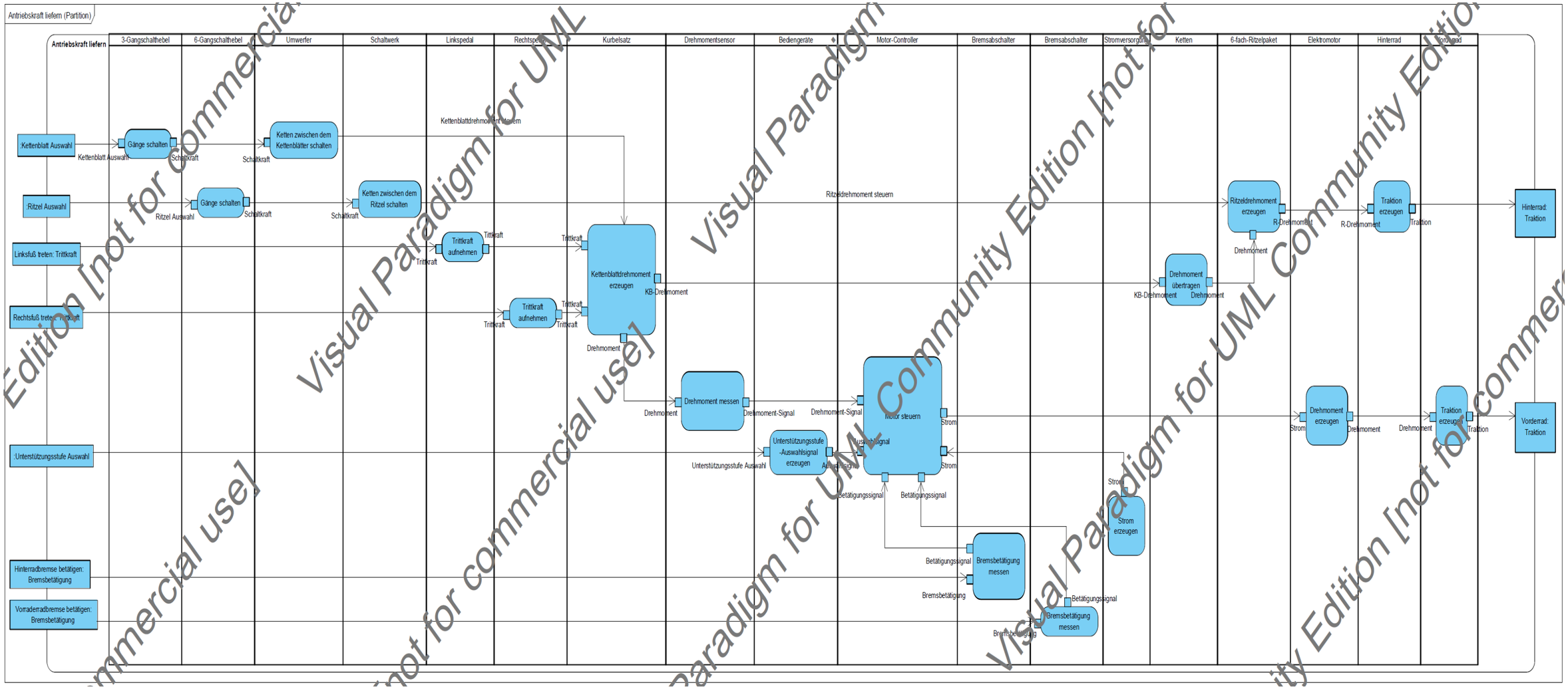

Abbildung 33: Aktivitätsdiagramm „Antriebskraft liefern“ mit den Partitionen

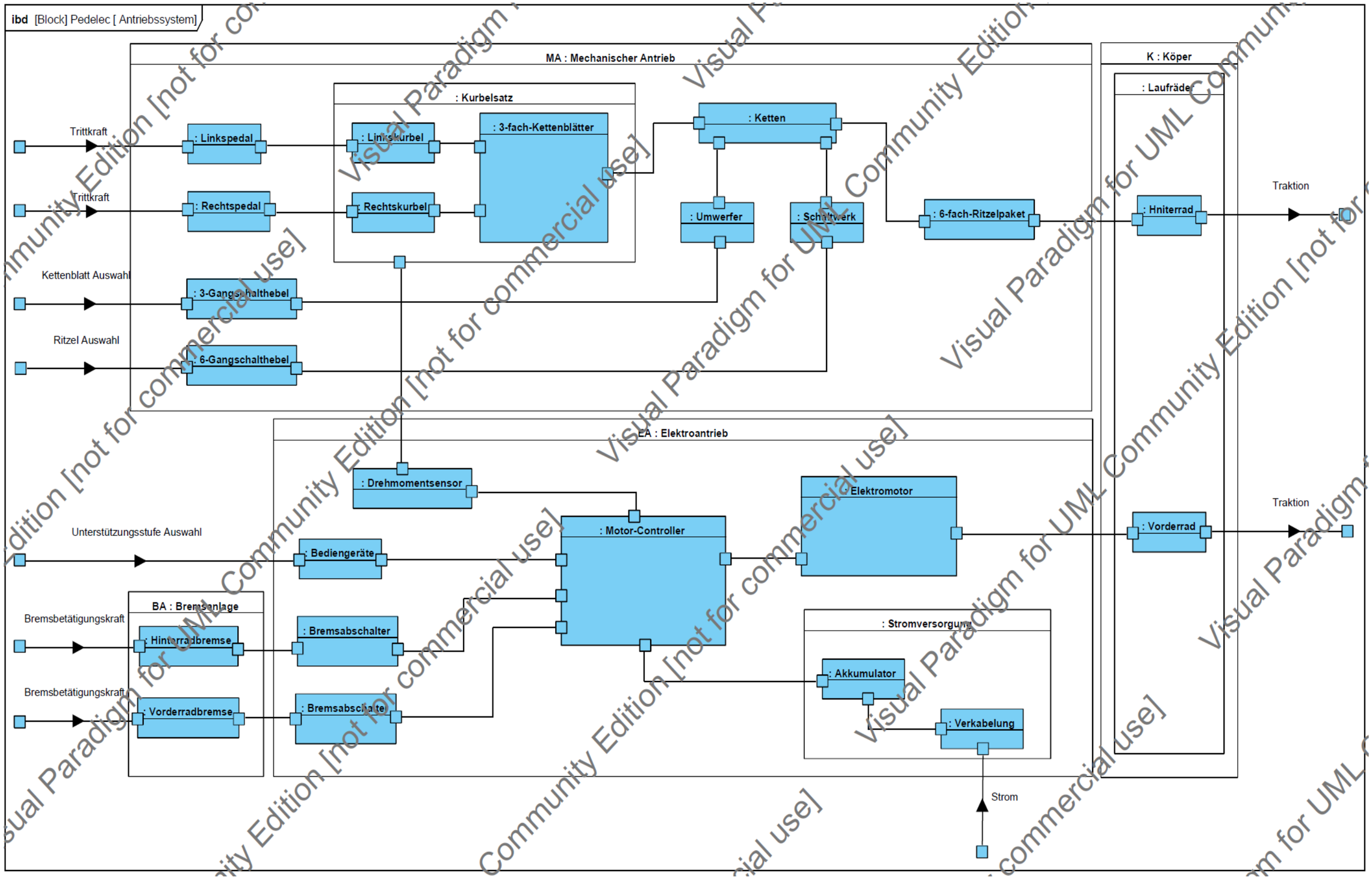

Abbildung 34: internes Blockdiagramm vom „Antriebssystem“

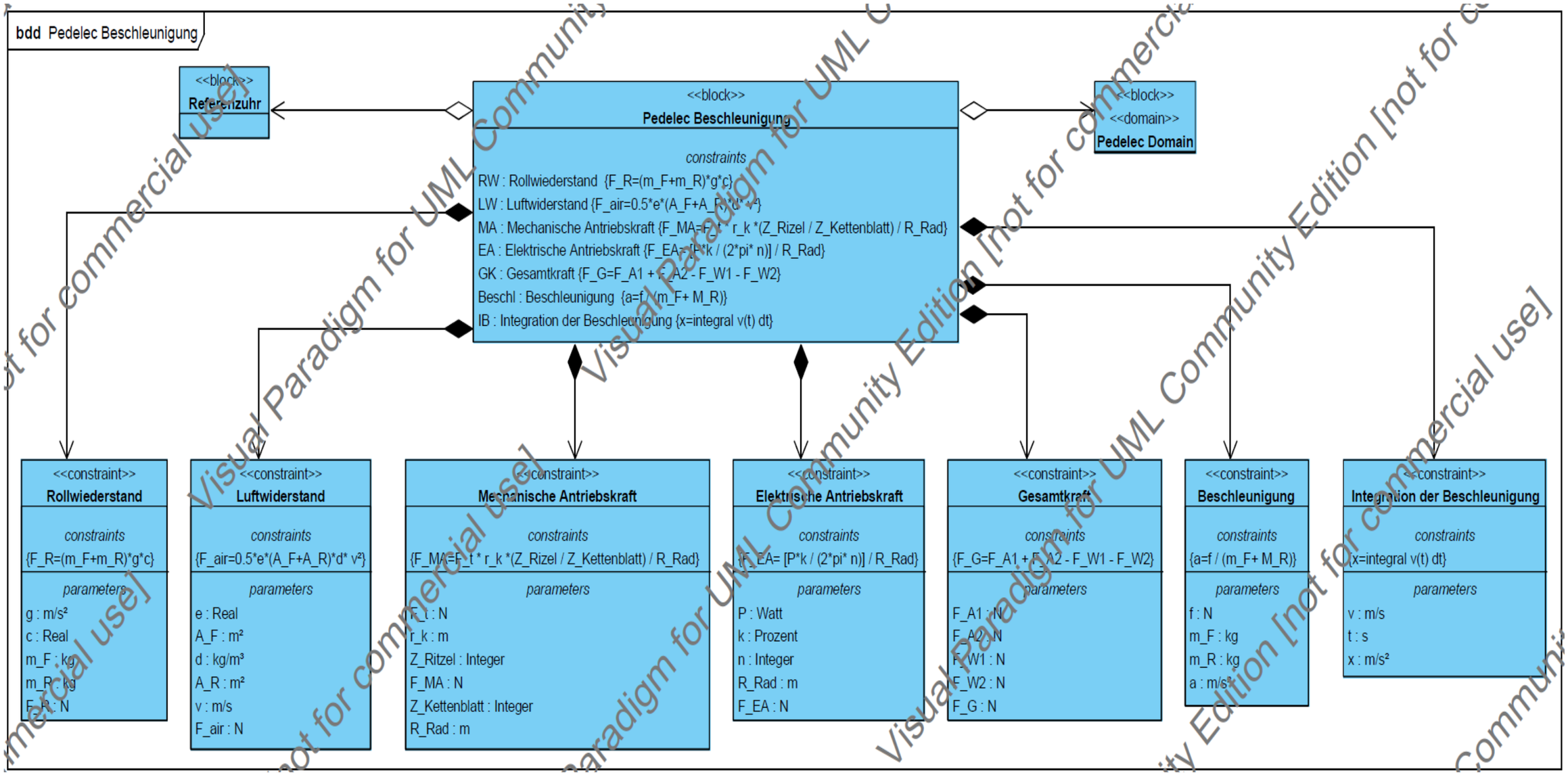

Abbildung 35: Blockdefinitionsdiagramm von „Pedelec Beschleunigung“

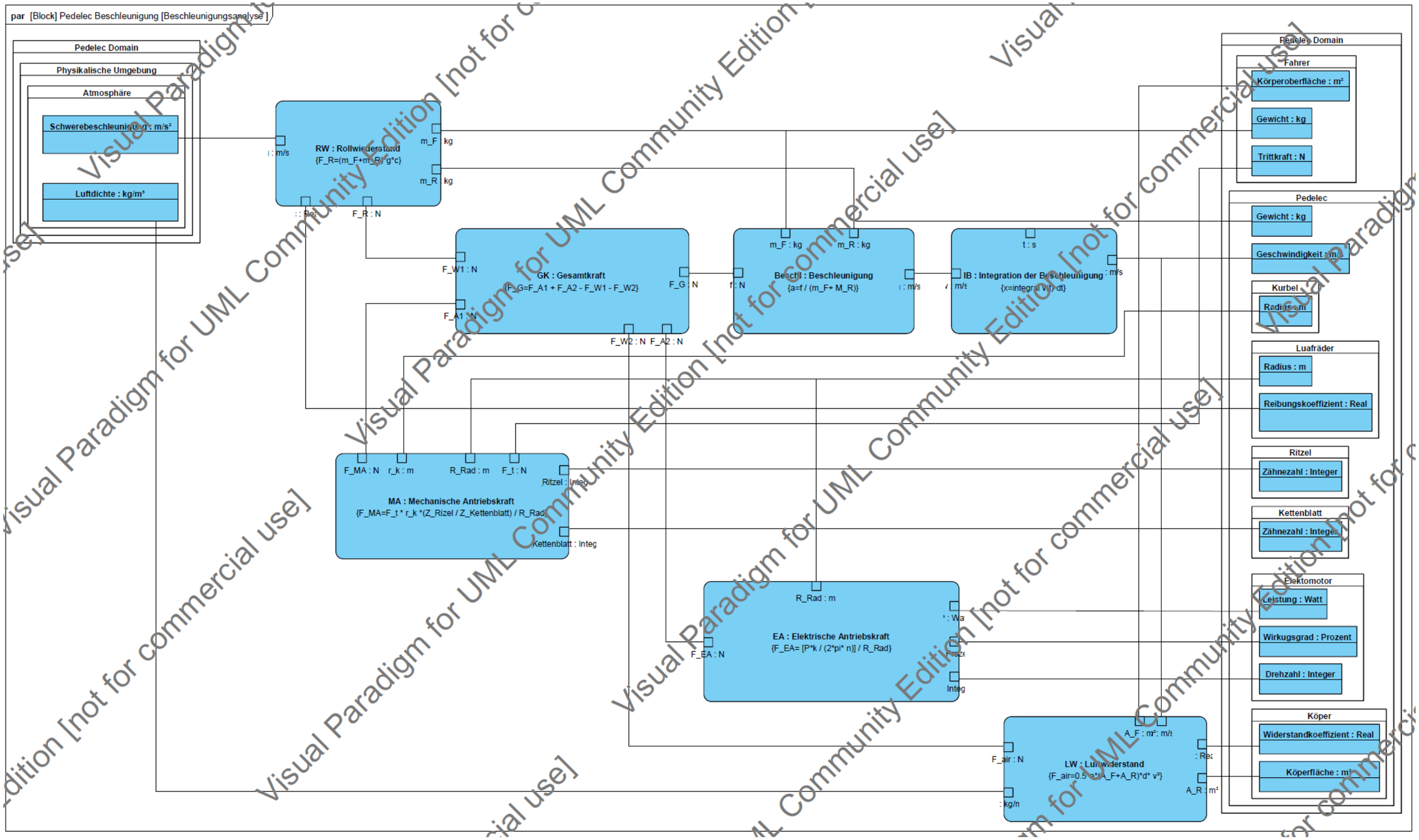

Abbildung 36: Parameterdiagramm von „Pedelec Beschleunigungsanalyse“

Literaturverzeichnis

B.Vogel-Heuser: Systems Software Engineering, Angewandte Methoden des Systementwurfs für Ingenieure, Oldenbourg Industrieverlag, München, 2003

Chris Paredis: Why Model-Based Systems Engineering? Benefits and Payoffs, (2008-2011) http://vpe.mv.uni-kl.de/cms/fileadmin/user_upload/PDF/PLM_Future_2012/02_PLM-Future-2012_Paredis.pdf (09.04.2013)

Daniel Steffen: Ein Verfahren zur Produktstrukturierung für fortgeschrittene mechatronische Systeme, Dissertation, Universtiät Paderborn, 2007

Edward A.Ladzinski: Collaborative Systems Engineering for the Industrial Equipment Industry, 2011. http://www.3ds.com/fileadmin/COMPANY/EVENTS/DSCC_2011/PDF/NOV8_435PM_IE_Ladzinski_V3.pdf (18.05.2013)

Eigner; Gilz; Zafirov: Interdisziplinäre Produktentwicklung – Modellbasiertes Systems Engineering. http://www.plmportal.org/forschung-details/items/interdisziplinaere-produktentwicklung-modellbasiertes-systems-engineering.html (10.04.2013)

Ernst Sikora: Ein modellbasierter Ansatz zur verzahnten Entwicklung von Anforderungen und Architektur über mehrere Abstraktionsstufen hinweg, Logos Verlag Berlin, 2010

Fahmi Bellalouna: Integrationsplattform für eine interdisziplinäre Entwicklung mechatronischer Produkte, Dissertation, Ruhr Universität Bochum, 2009

Feng Liang: Modeling in Modelica and SysML of System Engineering at Scania appliede to Fuel Level Display, Final Thesis, Linköpings universitet, 2012

Friedenthal; Griego; Sampson: INCOSE Model Based Systems Engineering (MBSE) Initiative, (2007). http://www.incose.org/enchantment/docs/07docs/07jul_4mbseroadmap.pdf (12.04.2013)

Frank Truyen: The Fast Guide to Model Driven Architecture, The Basis of Model Driven Architecture, Cephas Consulting Corp, January 2010. http://www.enterprisemodelingsolutions.com/public_ftp/Cephas_MDA_Fast_Guide.pdf (19.04.2013)

G.Pahl; W.Beitz; J.Feldhusen; K.-H.Grote: Konstruktionslehre, 7.Auflage, Springer-Verlag Berlin Heidelberg, 2007

INCOSE, Systems Engineering Vision 2020, INCOSE-TP-2004-004-02, Version 2.03, International Council on systems Engineering, September 2007.

Jeff A.Estefan: Survey of Model-Based Systems Engineering (MBSE) Methodologies, Pasadena, California, U.S.A. May 23.2008. http://omgsysml.org/MBSE_Methodology_Survey_RevB.pdf (19.04.2013)

Johannes Steinschaden: Lehrgang Konstruktionsmethodik, Fachhochschul Studiengänge Vorarlberg GmbH, Dornbirn, 1998/1999

Kogent Learning Solutions Inc: CATIA V6 Essentials, Jones and Bartlett Publischers, 2009.

M.Otter, C.Schweiger. Modellierung mechatronischer Systeme mit Modelica. http://elib.dlr.de/12172/1/otter2004-vdi-modelica.pdf (04.06.2013)

Michael Eckrich: Methodische Unterstützung zur Spezifikation, Validierung und Diagnoseentwicklung beim Entwurf mechatronischer systeme, Dissertation, Technischen Universität München, 1996

Matthias Gehrke: Entwurf mechatronischer Systeme auf Basis von Funktionshierarchien und Systemstruktuen, Dissertation, Universität Paderborn, 2005

Manuel Niekamp: Qualitätssicherung bei der Systementwicklung mit der SysML, Seminararbeit, Universität Paderborn, 2009.

Marco Pohl, Christian Scheel: Entwicklung verteilter eingebetteter Systeme, 2004. http://swt.cs.tu-berlin.de/lehre/eks/ws0304/SlidesPaper/ausarbeitung%20reqtify.pdf (26.05.2013)

Oliver Alt: Modellbasierte Systementwicklung mit SysML, Carl Hanser Verlag, München, 2012

Peer Hausding: Systemmodellierung mit SysML, Studienarbeit, Humboldt- Universität zu Berlin, 9,März 2010.

Parham Vasaiely: Interactive Simulation of SysML Models using Modelica, Bachelor Thesis, Hochschule für Angewandte Wissenschaften Hamburg, 2009.

Stefanie Constanze Zirkler: Transdisziplinäres Zielkostenmanagement komplexer mechatronischer Produkte, Dissertation, Technishen Universität München, 2010

Sanford Friedenthal, Alan Moore, Rick Steiner: A practical Guide to SysML, The Systems Modeling Language, Morgan Kaufmann, Waltham USA, 2012.

Sanford Friedenthal: Model-Based Systems Engineering (MBSE) 2012. http://www.lotustech.co.kr/PHX_forum2012/MC/SystemEngineering/Model-based%20Systems%20Engineering%20with%20SysML_Friedenthal.pdf (22.04.2013)

Sebastia J.I. Herzig, Markus Brandstätter: Übertragung von Methodiken des Software Engineering auf Model-Based System Engineering, in: Maik Maurer, Seven-Olaf Schulze (Hrsg.): Tag des Systems Engineering, Carl Hanser Verlag, München, 2010, S119ff.

Tim Weilkiens: Systems Engineering mit SysML/UML, Modellierung, Analyse, Design, 2.Auflage, dpunkt.Verlag, Heidelberg, 2006

VDI-Richtlinien 2206, Entwicklungsmethodik für mechatronische Systeme, Verein Deutscher Ingenieure, 2004

Winfried Dohmen: Interdisziplinäre Methoden für die integrierte Entwicklung komplexer mechatronischer Systeme, Dissertation, Technischen Universität München, 2002

Weilkiens: Zukunftsdisziplin Modellbasiertes Systems Engineering (19.-20.05.2011) http://www.conimit.de/uploads/tx_vitramemberadmin/literature/Zukunftsdisziplin-Systems-Engineering.weilkiens.oose.2011.pdf (13.04.2013)

Internetverzeichnis

http://www.eco2l.de/planung/systematik.html (10.03.2013)

http://de.wikipedia.org/wiki/Komplexit%C3%A4t#cite_note-4 (10.03.2013)

http://www.transmechatronic.de/start/mechatronik/?tx_felogin_pi1%5Bforgot%5D=1 (27.03.2013)

https://files.ifi.uzh.ch/rerg/arvo/ftp/ses/kapitel_12.pdf (28.03.2013)

http://www.xing.com/net/syng/modellbasiertes-systems-engineering-475034/was-ist-mbse-33792101/p10 (09.04.2013)

http://www.xing.com/net/syng/modellbasiertes-systems-engineering-475034/was-ist-mbse-33792101/p10 (09.04.2013)

http://www.incoseonline.org.uk/Program_Files/Publications/zGuides_9.aspx?CatID=Publications (12.04.2013)

http://www.xing.com/net/syng/modellbasiertes-systems-engineering-475034/was-ist-mbse-33792101/p0 (22.04.2013)

http://www.omgsysml.org/ (23.04.2013)

http://www.3ds.com/de/products/catia/portfolio/catia-version-6/overview/ (07.05.2013)

https://www.cenit.de/de/plm/lsungenprodukte/catia/catia-v6.html (14.05.2013)

http://www.claytex.com/products/catia-systems/ (15.05 2013)

http://www.plmportal.org/was-hat-systems-engineering-mit-plm-zu-tun.html (18.05.2013)

https://www.cenit.de/de/plm/lsungenprodukte/enovia/enovia-v6.html (19.05.2013)

https://www.cenit.de/de/plm/lsungenprodukte/enovia/enovia-v6/dokumentenmanagement.html (19.05.2013)

http://en.wikipedia.org/wiki/Requirements_management (20.05.2013)

http://www.claytex.com/products/catia-systems/requirements-engineering/ (20.05.2013)

http://www.3ds.com/products/enovia/products/enovia-v6/v6-portfolio/d/collaborative-innovation/s/governance-user/p/requirements-central/?cHash=f957afda68bb8007a08945ded16af2e8 (25.05.2013)

http://www.em.ag/themenloesungen/systems-engineering (25.05.2013)

http://www.claytex.com/products/reqtify/interfaces/ (26.05.2013)

http://www.3ds.com/de/products/catia/solutions/catia-systems-architecture/ (28.05.2013)

http://www.em.ag/produkte/catia-systems/ (28.05.2013)

http://www.plmportal.org/wo-steht-dassault-systemes.html (06.07.2013)

Collaborative Mechatronic Solution http://www.3ds.com/fileadmin/Industries/High-Tech/Pdf/solution-briefs/collaborative-mechatronics- engineering-high-tech-solution-brief.pdf (10.07.2013)

Abbildungsverzeichnis

Abkürzungsverzeichnis

AMS	Autonomes mechatronisches System
bzw.	beziehungsweise
CAE	Computer Aided Engineering
CAT	Computer Aided Testing
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CATIA	Computer Graphics-Aided Three-Dimensional Interactive Application
CIM	Computation Independent Model
d.h.	das heißt
DBSE	Dokumentenbasiertes Systems Engineering
etc.	et cetera
z.B.	zum Beispiel
MFM	Mechatronisches Funktionsmodul
MBSE	Modellbasiertes Systems Engineering
MDA	Model-Driven Architecture
MOF	Meta Object Facility
OMG	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
PLM	Product lifecycle Management
SE	Systems Engineering
SysML	System Modelling Language
UML	Unified Modeling Language
VMS	Vernetztes mechatronisches System