Herausforderungen und Anforderungen bei der durchgängigen Architekturmodellierung mechatronischer Systeme

Challenges and requirements for integrated architecture modeling of mechatronic systems

Thomas Schumacher[1]*, David Inkermann[1]

[1] Institute of Mechanical Engineering, Technische Universitaet Clausthal

** Korrespondierender Autor:*
Thomas Schumacher
Fritz-Süchting-Institut für Maschinenwesen der Technischen Universität Clausthal
Robert-Koch-Str. 32
D-38678 Clausthal-Zellerfeld
Telefon: +495323/725120
Mail: schumacher@imw.tu-clausthal.de

Abstract

Architectural models serve to generate a cross-domain understanding of system behaviour and relationships between subsystems within the development process. Various architectural models are described in literature and used in practice, differing in the level of concretion as well as the modelling purpose and representation. In order to structure these architectural models, a framework is derived from established and modern development models. The analysis of the architectural models using the model morphology indicates that architectural models have a large diversity and are often domain-specific. Consequently, the need of cross-domain architecture models are derived. Based on the analysis result and specific use cases requirements for cross-domain architecture modelling are defined.

Keywords

Model-based Systems Engineering, cross-domain architectural model, product architecture, model morphology, RFLP

 | https://doi.org/10.35199/dfx2020.6

1. Einleitung und Motivation

Der überwiegende Teil der Verhaltens- und Wirkeigenschaften von Systemen ist strukturabhängig. Um diese Eigenschaften bei der Entwicklung mechatronischer Systeme beurteilen und beeinflussen zu können, müssen domänenübergreifende Wirkzusammenhänge erkannt und abgebildet werden. Vorgehensweisen und Methoden des Systems Engineering (SE) stellen daher das domänenübergreifende Systemverständnis in den Fokus der Entwicklung. Hierzu werden Teilprozesse für die Systemarchitekturdefinition, z.B. ISO 15288, definiert und zunehmend formale Modelle in der Systementwicklung eingesetzt. Ziel des Model-based Systems Engineerings (MBSE) ist es, die Durchgängigkeit von Systembeschreibungen unterschiedlicher Konkretisierungsebenen und die Konsistenz verschiedener Sichten auf das zu entwickelnde System sicherzustellen. In diesem Beitrag werden zentrale Herausforderungen und Anforderungen der Architekturmodellierung mechatronischer Systeme analysiert.

1.1. Aufgabe und Bedeutung der Architekturmodellierung in der Systementwicklung

Um die aus der zunehmenden Vernetzung von Funktionen und Teilsystemen resultierende Komplexität moderner mechatronischer Systeme effizient handhaben zu können, sind Modelle zwingend erforderlich. Architekturmodelle beschreiben nicht nur die Anzahl und Art der Relationen zwischen Elementen, sondern definieren auch topologische Eigenschaften wie bspw. Reihenfolge oder Redundanz von Funktionen und Systemelementen, siehe Abschnitt 2.1. Im Entwicklungsprozess werden diese Architekturmodelle eingesetzt, um bspw. Systemeigenschaften zu definieren und zu überprüfen, Teilergebnisse zu integrieren, Entwicklungsprozesse und -aktivitäten zu koordinieren oder Maßnahmen für spätere Lebenslaufphasen zu planen [1]. Konkrete Aufgaben von Architekturmodellen können bspw. für die Beurteilung der Einflüsse von Änderungen einzelner Systemkomponenten oder -funktionen (Impact Analysis) und Bewerten der Zuverlässigkeit von Systemen, die Definition von Referenzarchitekturen sowie Planung von Modulen für die effiziente Entwicklung, Integration und Realisierung von Systemen (Entwicklung modularer Produktstrukturen) oder die Erstellung von Workbreak-Down Structures zur Koordination von Entwicklungsprojekten sein. Diese exemplarischen Anwendungsfälle verdeutlichen die zentrale Stellung von Architekturmodellen in der Systementwicklung sowie Projektkoordination und zeigen gleichzeitig unterschiedliche Schwerpunkte auf.

1.2. Zielsetzung und Struktur des Beitrags

Ziel des Beitrags ist es, ein grundlegendes Verständnis der Herausforderungen und Anforderungen der Architekturmodellierung in der Systementwicklung zu beschreiben. Hierbei wird folgende Forschungsfrage fokussiert: Welcher methodische Ansatz eignet sich zur Strukturierung von Architekturmodellen, für welche Anwendungsfälle werden Architekturmodelle eingesetzt und welche Anforderungen an domänenübergreifende Architekturmodelle resultieren hieraus? Zur Beantwortung umfasst dieser Beitrag fünf Kapitel. Im zweiten Kapitel werden zunächst zentrale Begriffe definiert und Grundlagen des SE und MBSE eingeführt. Im dritten Kapitel wird ausgehend von etablierten Vorgehensweisen und Ansätzen zur Gliederung von Modellen im Entwicklungsprozess ein Rahmenwerk für die Strukturierung von Architekturmodellen vorgeschlagen. Mithilfe einer Modellmorphologie werden im vierten Kapitel wesentliche Unterschiede häufig verwendeter Architekturmodelle analysiert und hieraus Anforderungen an eine domänenübergreifende Modellierungsmethodik abgeleitet. Die Diskussion und ein Ausblick schließen den Beitrag ab.

2. Grundlagen Architekturmodelle und Model-based Systems Engineering

In diesem Kapitel wird das Verständnis von Architekturmodellen eingeführt und zu etablierten Begriffen abgegrenzt. Zudem werden Grundlagen und Ziele des SE und MBSE erläutert.

2.1. Verständnis und Einordnung des Begriffs Architekturmodell

In der Literatur finden sich zahlreiche Definitionen der Begriffe (System-)Architektur oder Produktarchitektur. Dabei werden die Begriffe Struktur und Architektur vielfach synonym verwendet, bspw. [2]. In der Produktentwicklung wird der Begriff Struktur als „Gegliederter Aufbau, innere Gliederung, Gefüge, das aus Teilen besteht, die wechselweise voneinander abhängen“ beschrieben [3] und üblicherweise zwischen Funktions-, Wirk- und Baustrukturen (Produktstrukturen) unterschieden, siehe Abschnitte 3.1 und 4.2. Das in der Produktentwicklung etablierte Konzept der Produktarchitektur [2] stellt die Verknüpfung der funktionalen Produktbeschreibung (Funktionsstruktur) und dem physischen Aufbau des Produktes (Produktstruktur) in den Vordergrund, siehe Bild 1.

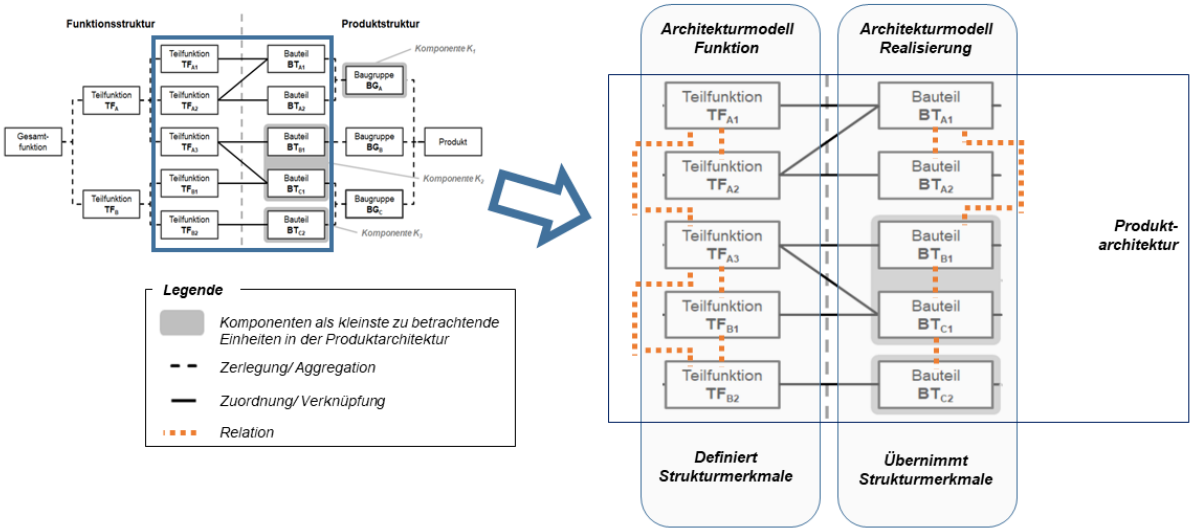

Bild 1: Zusammenhang zwischen Architekturmodellen und Produktarchitektur, basierend auf [4, 5]

Die Verknüpfung von Funktionen und Bauteilen wird hierbei fokussiert, um unterschiedliche Bauweisen (Konstruktionsprinzipien) zur Realisierung der Produktstruktur zu unterscheiden und produktstrategische sowie organisatorische Aspekte bei der Produktstrukturierung zu berücksichtigen [4, 5]. Dieses Verständnis soll durch die Einführung des Begriffes Architekturmodell erweitert werden. Unter Architekturmodellen werden demnach Strukturbeschreibungen verstanden, durch die Anordnungen, Reihenfolgen und funktionale Beziehungen zwischen einer Menge von Elementen gleicher oder unterschiedlicher Konkretisierungsebenen definiert werden. Architekturmodelle werden im Entwicklungsprozess auf unterschiedlichen Konkretisierungsebenen verwendet und dokumentieren die jeweils erforderlichen Festlegungen von Strukturmerkmalen des zu entwickelnden Systems. Die aus den Strukturmerkmalen der Architekturmodelle unterschiedlicher Konkretisierungsebene hervorgehende Struktur entspricht der Produktarchitektur, siehe Bild 1. Diese Erweiterung ist zweckmäßig, da durch die Architekturmodelle auf den unterschiedlichen Konkretisierungsebenen wesentliche Strukturmerkmale [6] definiert werden. Bspw. hat die Reihenfolge oder Redundanz von Funktionen innerhalb der Funktionsstruktur maßgeblichen Einfluss auf die Auswahl und Auslegung von Effekten und Effektträgern. Für die Softwarerealisierung sowie Auswahl und Auslegung von Sensoren ist es zudem entscheidend, ob die Erfassung und Verarbeitung von Messgrößen und Signalen bspw. zeitgleich oder nacheinander erfolgt. Gleichzeitig müssen diese topologischen Festlegungen im Sinne einer durchgängigen Modellkette über die Konkretisierungsebenen hinweg konsistent sein. In der praktischen Entwicklungsarbeit werden verschiedenartigste Architekturmodelle eingesetzt, siehe Abschnitt 4.2. Die in der Domäne der Mechanik verwendete Produktstruktur stellt hierbei eine spezifische Klasse von Architekturmodellen dar.

2.2. Systems Engineering und Model-based Systems Engineering

SE und MBSE werden branchenübergreifend als vielversprechende Ansätze zur Handhabung steigender Produkt- und Entwicklungskomplexität angesehen [7, 8]. Zentrale Grundlagen des SE sind die allgemeine Systemtheorie [9] und die allgemeine Modelltheorie [10]. Haberfellner et al. [11] betonen die Notwendigkeit, Methode und Werkzeuge aus beiden Bereichen zu integrieren und gleichzeitig eine SE-Denkweise zugrunde zulegen. Diese Denkweise impliziert zwei Grundprinzipien: das *System-of-System-Prinzip* (SoS) und das *System-of-Interest-Prinzip* (SoI). Das SoS-Prinzip definiert, dass komplexe Systemkomponenten selbst als Systeme betrachtet werden müssen. Das SoI-Prinzip hebt hervor, dass das zu entwickelnde System immer in eine Umgebung eingebettet ist und mit anderen Systemen dieser Umgebung interagiert und sich an diese anpasst. Die Festlegung des SoI stellt immer eine Vereinfachung der Realität und damit meist eine Reduktion der Komplexität dar, indem Elemente und Beziehungen hervorgehoben, die für die aktuelle Entwicklungsaufgabe relevant sind. MBSE baut auf diesen Prinzipien auf und stellt die konsequente Verwendung von (formalen) Modellen in den Vordergrund. Übergeordnetes Ziel ist es, die Ergebnisse unterschiedlicher Entwicklungsaktivitäten in einem Modell zusammenzuführen. Die Systemmodellierung umfasst dabei die Elementen Sprache, Tool und Methodik [12, 13]. Durch die Sprachstandards (bspw. SyML, UML) und die Tools können Sichten auf die Modelle realisiert werden, die jeweils an spezifische Entwicklungsaktivitäten angepasst sind. Änderungen innerhalb dieser Sichten werden unmittelbar mit dem zugrundeliegenden Modell abgeglichen und damit die Konsistenz sichergestellt. Die Festlegung einer initialen Systemarchitektur (Architecture Baseline) wird in MBSE-Methodiken in der Regel als Ergebnis der Systemanalyse und des Systementwurfs vorgeschlagen und umfasst Festlegungen für alle Entwicklungsdomänen [14], (ISO 15288).

3. Rahmenwerk für die Strukturierung und Verknüpfung von Architekturmodellen

In diesem Kapitel wird ein Rahmenwerk für die Strukturierung und Verknüpfung von Architekturmodellen eingeführt. Das Rahmenwerk gliedert sich in Anforderungs- und Lösungsraum mit den Partialräumen *Funktionsraum*, *Logischer Raum* und *Realisierungsraum*.

3.1. Bestehende Ansätze zur Strukturierung von Modellen in der Systementwicklung

Forschungsarbeiten in der Produktentwicklung und Softwareentwicklung haben unzählige Ansätze zur Strukturierung produktbeschreibender Modelle hervorgebracht. Trotz unterschiedlicher Begrifflichkeiten und Darstellungen, ist der Konkretisierungsgrad ein wiederkehrendes Merkmal zur Strukturierung der Modelle. Der auf Rude [15] zurückgehende Modellraum des Konstruierens greift die Phasen des Entwicklungsprozesses nach Pahl & Beitz [3] auf und definiert Anforderungen, Funktionen, Prinzipien und Gestalt als Gliederungsebenen. Neben der Konkretisierung/ Abstraktion werden die Zerlegung/ Zusammenführung und die Variation/ Einschränkungen als Dimensionen des Modellraums zugrunde gelegt. Das Münchner Konkretisierungsmodell greift diese Dimensionen auf und hebt die Unterscheidung zwischen Anforderungs- und Lösungsraum hervor. Der separate Anforderungsraum verdeutlicht hierbei, dass Anforderungen über den gesamten Entwicklungsprozess eingesteuert, erweitert, gepflegt und überprüft werden müssen [16]. In weiteren Ansätze wie dem RFLP-Ansatz [17, 18], dem SPES Modeling Framework [19] und der aktualisierten VDI 2221 [21] werden mit variierender Terminologie vier Sichtweisen (SPES) oder Ebenen (RFLP) der Systemmodellierung unterschieden: *Anforderungsebene*, *Funktionsebene*, *Logische Ebene* und *Physikalische/ Technische Ebene*. Auf der Anforderungsebene erfolgt die Definition und Entwicklung der Produktanforderungen um die erforderlichen Produkteigenschaften festzulegen und um den geforderten Verwendungszweck zu erfüllen [18, 19]. Die Funktionsebene umfasst die funktionale Architektur des Systems, welche möglichst lösungsneutral die Anordnung und Verknüpfung einzelner (Teil)Funktionen beschreibt [19, 20]. Die logische Ebene definiert die logische Architektur des

zu entwickelnden Systems durch Unterteilung des Systems in kommunizierende Komponenten (Software) [19] oder Festlegung und Verknüpfung von Wirkprinzipien (Mechanik) [18, 19]. Auf der physikalischen Ebene werden die konkreten Realisierungen der Wirkprinzipien durch Bauteile definiert und spezifiziert Hardwareressourcen auf denen die Software ausgeführt wird.

3.2. Rahmenwerk für die Strukturierung von Architekturmodellen

Auf Grundlage der analysierten allgemeinen Ansätze zur Strukturierung von Modellen in der Systementwicklung wurde das in Bild 2 dargestellte, erweiterte Rahmenwerk für die Unterscheidung und Einordnung von Architekturmodellen abgeleitet. Dieses greift die zuvor begründete Unterscheidung von Anforderungs- und Lösungsraum auf und unterteilt den Lösungsraum analog der zuvor identifizierten Gliederungsebenen in die drei Partialräume *Funktionsraum*, *Logischer Raum* und *Realisierungsraum*. Die räumliche Ausdehnung der Partialräume verdeutlicht dabei, dass Architekturmodelle gleichen Zwecks bspw. Abbildung der Funktion in unterschiedlichen Konkretisierungsgraden vorliegen können. Ein Beispiel hierfür ist die Bestimmung der relevanten Use Cases mit schrittweiser Detaillierung durch Aktivitäten (Aktivitätsdiagramme) innerhalb des Funktionsraums. In dem Bild wurden drei schematische Architekturmodelle integriert. Typische Modelle zur Beschreibung der funktionalen Systemarchitektur sind u.a. SysML/ UML Aktivitätsdiagramme, die eine Abfolge von Aktivitäten sowie erforderliche Eingang- und Ausgangsinformationen spezifizieren [13]. Diese Diagrammtypen werden vermehrt neben der Softwareentwicklung auch in der mechatronischen Systementwicklung eingesetzt. Die in der funktionalen Architektur definierten Strukturmerkmale werden in den logischen Lösungsraum übergeben. In dem Bild 2 wurden die Strukturmerkmale aus den Aktivitätsdiagramm in ein Internes Blockdiagramm überführt. Interne Block Diagramme erlauben die Modellierung der Systems- bzw. Subsystemstruktur einschließlich der Relationen (Fluss- und Kontrollschnittstellen). Basierend auf der logischen Architektur wird im Realisierungsraum die physikalische Architektur des Produktes bestimmt, dies kann bspw. durch 3D-CAD-Modellen erfolgen.

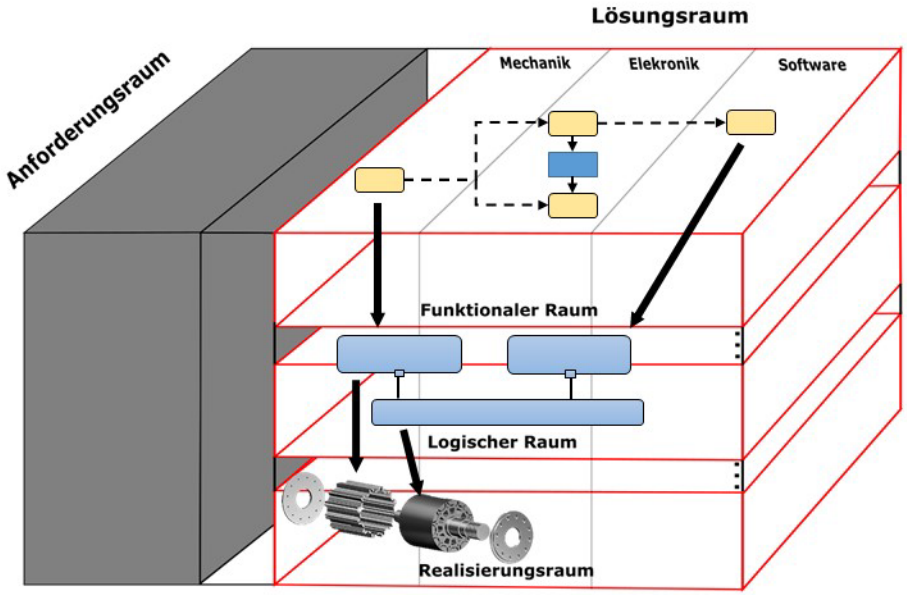

Bild 2: Rahmenwerk für die Strukturierung von Architekturmodellen, basierend auf [16]

Bild 2 verdeutlicht, dass sowohl innerhalb der drei eingeführten Partialräume als auch zwischen den Partialräumen Schnittstellen bestehen. Beispielsweise können zwischen der mechanischen und der elektronischen Domäne physikalische Verbindungen, wie Verschraubun-

gen, als Schnittstellen existieren. Die Schnittstellen zwischen der Elektronik- und der Softwaredomäne ist durch Informations- und Steuerungssignale charakterisiert [21]. Die Einführung domänenübergreifender Architekturmodelle innerhalb der partialen Lösungsräume erhöht das Gesamtsystemverständnis und die Transparenz von Wirkzusammenhängen zwischen den Teilsystemen.

4. Analyse von Anwendungsfällen und Architekturmodellen

In diesem Kapitel werden praxisrelevante Anwendungsfälle für Architekturmodelle erläutert und auf Grundlage des zuvor eingeführten Rahmenwerks ausgewählte Architekturmodelle aus Literatur und Praxis analysiert.

4.1. Typische Anwendungsfälle von Architekturmodellen in der Entwicklungspraxis

Anwendungsfälle werden in unterschiedlichen, teilweise domänenübergreifenden Anwendungsfällen in der Entwicklungspraxis eingesetzt. Auf Basis von Interviews mit Industriepartnern sowie Erfahrungen in der Bearbeitung anwendungsnaher Entwicklungsprojekte können folgende Anwendungsfälle, siehe Tabelle 1, als häufig und repräsentativ eingestuft werden.

Tabelle 1: Anwendungsfälle von Architekturmodellen

Anwendungsfälle	Erläuterung
Wiederverwendung bestehenden Subsysteme	Einsatz von Architekturmodelle, um die Wiederverwendbarkeit von Subsystemen zu überprüfen und erforderliche Änderungen zu identifizieren. Bspw. kann durch Bewertung der funktionalen und logischen Architekturmodelle kann geprüft werden, ob eine bereits entwickelte Leistungselektronik in ein neues Antriebskonzept im Fahrzeug integrierbar ist. Dazu müssen neben den erforderlichen Funktionalitäten insbesondere die mechanischen, elektrischen und softwaretechnischen Schnittstellen überprüft werden.
Produktgenerationenentwicklung	Anpassung der Architekturmodelle durch gezielte Variation an neue Anforderungen oder an geänderte Umgebungsbedingungen. Bei der Prinzipvariation wird ein Produkt basierend auf der funktionalen Architektur durch Anpassung des Lösungsprinzips (logische Architektur) entwickelt. Bei der Gestaltvariation wird das Lösungsprinzip beibehalten und die physikalische Architektur (Realisierungsraum) angepasst. Aufgrund der Festlegung und Weitergabe von Strukturmerkmalen zwischen den partialen Lösungsräumen sollte der Funktionsraum auch immer betrachtet werden.

Diese Anwendungsfälle verdeutlichen einerseits unterschiedliche Schwerpunkte im Hinblick auf die eingeführten partialen Lösungsräume. Anderseits wird die Notwendigkeit der Verknüpfung der Architekturmodelle aus den unterschiedlichen Lösungsräumen deutlich.

4.2. Analyse ausgewählter Architekturmodelle

In der Literatur und Entwicklungspraxis existieren eine Vielzahl verschiedener Architekturmodelle. Um zentrale Unterschiede zwischen Architekturmodellen verschiedener Domänen zu ermitteln sowie die Verknüpfung von Architekturmodellen unterschiedlicher Konkretisierungsebenen zu untersuchen, wurde eine strukturierte Analyse durchgeführt. Die Auswahl der Modelle erfolgte durch Betrachtung relevanter Fachbücher und im Gespräch mit Fachexperten. Für die Analyse wurde die Modellmorphologie nach Buur & Andreasen [22] adaptiert und angewendet. Tabelle 2 zeigt einen Ausschnitt der Analyse sowie die Verortung der Architekturmodelle in Funktionsraum, logischem Raum und Realisierungsraum. Die durchgeführte Analyse zeigt deutlich die Heterogenität und die domänenspezifische Anwendung von Architekturmodellen. Die folgenden Abschnitte fassen die wesentlichen Erkenntnisse zusammen.

Tabelle 2: Ausschnitt der Analyse und Strukturierung bestehender Architekturmodelle

	Modell	Gegenstand	Eigenschaften	Zweck	Darstellung	Domäne
F *(Funktionsraum)*	Funktionsstruktur, *Pahl&Beitz [3]*	Erforderliche (Teil)Funktionen und Zusammenhänge	Haupt- und Teilfunktionen (verbal beschrieben), Energie-, Stoff-, Materialflüsse	Definition Teilaufgaben für weitere Entwicklung, initiale Schnittstellendefinition	Funktionsboxen, Flüsse, textuelle Beschreibungen	Mechanik
	Aktivitätsdiagramm, *SysML/ UML [13]*	Erforderliche Aktivitäten und logische Abhängigkeiten	Aktionen (verbal beschrieben), Knoten, Kontroll- und Informationsflüsse	Festlegung erforderliche Aktivitäten inkl. Abfolgen & Abhängigkeiten, ggf. Parallelität	Symbole und Text, standardisiert nach UML	Software
	Funktionslisten, *Praxis*	Erforderliche/ verfügbare Teilfunktionen eines Systems	textuelle Beschreibung, ggf. Hierarchische Gliederung	Übersicht verfügbare/ im Projekt genutzter (Teil)Funktionen	Textuelle Beschreibung, hierarchische Gliederung	Mechatronik
L *(Logischer Raum)*	Wirkstrukturmodelle, *Pahl&Beitz [3]*	Effektträger zur Realisierung von (Teil)Funktionen	Anordnung und Verknüpfung von Effektträgern, grobe Eigenschaften und relevante Merkmale	Definition erforderlicher Effektträger, Anordnungsstudien, Konzeptvergleiche	Prinzipskizzen, Symbole für Effektträger (nicht standardisiert)	Mechanik
	(Internal) Block Definition Diagramm, *SysML/ UML [13]*	(Software-) Komponentenstruktur & Kommunikation	Komponenten, Ports, Konnektoren, Aggregationen, Benennungen	Schnittstellen Definition zwischen (virtuellen) Komponenten, Modularisierung	Symbole und Texte, standardisiert nach UML	Software
	Nicht formalisierte Wirkstrukturmodelle *Praxis*	Komponentenstruktur, Kommunikation, Schnittstellen; angereichert mit spezifischen Informationen	Komponenten, Kommunikationspfade, Schnittstellen, Benennungen; spezifische Informationen wie Liefer- oder Entwicklungsumfänge	Darstellung der Systemstruktur und Schnittstellen; weitere spezifische Verwendungszwecke, bspw. Definition des Entwicklungsumfangs	Prinzipskizzen, Symbole, Gliederungen, textuelle Beschreibungen, Pfade, Verknüpfungen	Mechatronik
R *(Realisierungsraum)*	Baustrukturmodelle, *Pahl&Beitz [3]*	Bauteile, Verbindungen, Baugruppen, Gesamtsystem	Konkreter technischer System- und Systemelementeaufbau	Definition erforderlicher System- und Systemelemente, Bewertung von Fertigungs- bzw. Montageanforderungen	Prinzipskizzen, Zeichnung	Mechanik
	(Internal) Block Definition Diagramm, *SysML/ UML [13]*	Softwarekomponenten, Subsysteme, Implementierungstechnologie, Schnittstellen	Bezeichnung der Komponenten / Subsysteme, Variablen und Variablentypen, Funktionale Zusammenhänge, Annotationen	Festlegung aller relevanten Informationen zur automatisierten Quellcodeerstellung	Symbole, textuelle Beschreibungen, Pfeile, Linien, standardisiert nach UML	Software
	3D-Modelle, Stücklisten *Praxis*	Systemaufbau, Komponentenstruktur	Geometriedaten, Komponenten, Schnittstellen	Visualisierung des Systems, Bestimmung von Systemschnittstellen, Festlegung der Montagereihenfolge	Zeichnungen, dreidimensionale Bauteile, Werte, Beschreibungen	Mechatronik

4.2.1. Unterschiede in Formalisierung und Visualisierung

Formalisierung und Visualisierung betreffen die Darstellung der Architekturmodelle. Hierbei sind sowohl innerhalb der Entwicklungsdomänen, als auch zwischen den Domänen und den Konkretisierungsebenen erhebliche Unterschiede festzustellen. Neben rein textuellen Beschreibungen mit hierarchischer Gliederung (bspw. Funktionslisten in Tabellenform), existieren unterschiedlich stark formalisierte Ansätze für die Darstellung von Funktionsstrukturen in den Domänen der Mechanik und der Mechatronik, bspw. [23]. Hierbei erfolgt mit zunehmender Konkretisierung in der Regel eine stärkere Formalisierung durch vordefinierte Symbole. Einen starken Formalisierungsgrad weisen Aktivitätsdiagramme aufgrund der standardisierten Syntax (SysML/ UML) auf. Erhebliche Unterschiede bestehen in der Formalisierung von Wirkstrukturmodellen im logischen Raum. Neben klassischen Prinzipskizzen mit sehr geringem Formalisierungsgrad existieren stärker formalisierte Ansätze für die Modellierung. Blockdiagramme hingegen verwenden vordefinierte Symbole, ergänzt um beschreibende Texte. Dieser Unterschied ist auch für den Realisierungsraum gültig, Baustrukturmodelle der Mechanik sind erheblich weniger standardisiert als bspw. Plattformspezifische Modelle in der Softwareentwicklung. Die beschriebenen und in Tabelle 1 aufgeführten Architekturmodelle nutzen überwiegend

graphische Darstellungen (ausgenommen Funktionslisten und Stücklisten). Ergänzend werden in der Praxis vereinzelt matrixbasierte Darstellungen verwendet, um bspw. Relationen zwischen Funktionen anzugeben oder Funktionen und Elemente des logischen Raum zuzuordnen, siehe Abschnitt 4.2.3.

4.2.2. Unterschiede der abgebildeten Elemente und Relationen

Aufgabe von Architekturmodellen ist es, Strukturmerkmale zu definieren und zu dokumentieren. Die Analyse der hierzu in den Architekturmodellen abgebildeten Elemente zeigt ebenfalls große domänenspezifische Unterschiede. Während in der Mechanik überwiegend Energie- und Materialflüsse (Funktionsraum) sowie räumliche Anordnungen und Relativbewegungen (logischer Raum) abgebildet werden, werden in den Architekturmodellen der Softwareentwicklung insbesondere logische Abhängigkeiten und Reihenfolgen von Aktivitäten (Funktionsraum) abgebildet. Ein wesentlicher Unterschied besteht hinsichtlich der im logischen Raum verwendeten Elemente. In den Wirkstrukturmodellen der Mechanik entsprechen die logischen Elemente einzelnen Wirkprinzipien, d.h. physikalischen Effekten. Die logischen Elemente der Softwaredomäne hingegen entsprechen der Zerlegung des zu entwickelnden Systems in miteinander kommunizierende Komponenten [19]. Diese Zerlegung in realisierbare Komponenten erfolgt in der Mechanik in der Regel erst im Übergang zum Realisierungsraum. Die Architekturmodelle des Realisierungsraums der mechanischen Domäne, wie Baustrukturmodelle, zeigen den konkreten Systemaufbau durch Definition der Bauteile und Baugruppen und bspw. ihren kraft- und bewegungsübertragenden Verbindungen. In der Softwareentwicklung legt das Architekturmodell des Realisierungsraums die Softwarekomponenten und die Implementierungstechnologie fest, um anschließend möglichst automatisiert den Quellcode zu erzeugen.

4.2.3. Verknüpfungen zwischen den Lösungsräumen

Im Rahmen der Entwicklung mechatronischer Systeme sind insbesondere die Verknüpfungen zwischen den partialen Lösungsräumen relevant, da diese eine Übergabe von Strukturinformationen mit zunehmender Lösungskonkretisierung ermöglichen. Bei Betrachtung der Mechanik-Domäne wird deutlich, dass die einzelnen Architekturmodelle zwar aufeinander aufbauen, jedoch in der Regel keine formalisierten Verknüpfungen von Elementen der einzelnen Räume definiert werden. Diese häufig fehlende Verknüpfung ist auf eine polyhierarchische Vernetzung (n:m-Beziehung) mit den Elementen (Wirkprinzipien, Bauteilen) der nachfolgenden Ebene zurückzuführen. Dies erschwert die nachvollziehbare Übergabe von Strukturmerkmalen. Die Verknüpfung von Architekturmodellen in der Softwareentwicklung ist hingegen stärker formalisiert, auch um einen hohen Automatisierungsgrad in der Quellcode-Erzeugung zu erzielen. Die Verwendung einer standardisierten Modellsyntax sowie Dekompositions- und Verfeinerungsprinzipien unterstützt hierbei die Durchgängigkeit von Strukturmerkmalen. Gleichzeitig besteht auch in der Softwareentwicklung die Herausforderung der polyhierarchischen Vernetzung, der teilweise durch Vorgabe von Architekturprinzipien, wie bspw. 1:n Zuordnung im Übergang vom Funktionsraum in den logischen Raum [19]. Ein übliches Hilfsmittel zur Abbildung der Verknüpfungen zwischen Elemente der partialen Lösungsräume sind Matrizen (Design Structure Matrix, [24]). Wesentlicher Nachteil dieser Modellierung der Verknüpfungen ist, dass in der Regel ein weiteres rechnerunterstütztes Werkzeug genutzt werden muss. Einzelne Softwarelösungen wie Enterprise Architect bieten die Möglichkeit Matrizen aus SysML/ UML-Diagrammen abzuleiten und Änderungen in die Diagramme zurückzuspielen.

4.3. Anforderungen an eine domänenübergreifende Architekturmodellierung

Basierend auf der Analyse und den Herausforderungen lassen sich folgende Anforderungen ableiten, siehe Tabelle 3.

Tabelle 3: Anforderungen an die durchgängige Architekturmodellierung

	Wesentliche Anforderungen
1	Die Konsistenz von Strukturmerkmalen innerhalb domänenspezifischer Architekturmodelle muss sichergestellt werden
2	Domänenspezifische Schnittstellen müssen bei Verwendung von domänenspezifischer Architekturmodelle abgebildet werden können
3	Architekturmodelle müssen die effiziente Aggregation von Strukturinformationen unterstützen, um die Wiederverwendung bestehender Lösungselemente zu ermöglichen
4	Architekturmodelle müssen die Durchführung von Variationsoperationen zur effizienten Änderung von Strukturmerkmalen ermöglichen
5	Architekturmodelle sollten die einfache Visualisierung auf unterschiedlichen Konkretisierungsebenen und für unterschiedliche Stakeholder im Entwicklungsprozess unterstützen

Diese Anforderungen stellen grundsätzliche Forderungen zur Verbesserung der durchgängigen Modellierung von Architekturen im Entwicklungsprozess dar. Ergänzend ist die Trennung zwischen logischem Raum und Realisierungsraum eine zentrale Forderung, um die konzeptionellen Eigenschaften einer Lösung von technologischen Einschränkungen und Entwurfsentscheidungen zu trennen. Gleichzeitig muss bei der Konzeptausarbeitung berücksichtigt werden, dass während der Entwicklung häufig domänenspezifische Architekturmodelle unterschiedlicher Konkretisierungsebenen, bspw. Wirkstrukturmodelle der Mechanik und Aktivitätsmodelle für die Softwarelösung kombiniert werden. Ein möglicher Lösungsansatz ist das Konzept der heterogenen Modellierung [25]. Hierbei werden Architekturmodelle unterschiedlicher Konkretisierungsgrade in einem Systemmodell verknüpft. Auf diese Weise können domänenspezifische Eigenschaften und Restriktionen sowie unterschiedliche Ausarbeitungsstände abgebildet werden. Beispielsweise können existierende Baugruppen (3D-Modell) im Rahmen der Produktgenerationenentwicklung durch zusätzliche Funktionen zur Bewegungsführung oder –erfassung ergänzt werden, um das erforderliche Gesamtsystemverständnis abzuleiten.

5. Zusammenfassung und Ausblick

Dieser Beitrag beschreibt Herausforderungen und Anforderungen für die domänenübergreifende Architekturmodellierung mechatronischer Systeme. Hierzu wurden zunächst der Begriff Architekturmodell eingeführt und ein Rahmenwerk für die Strukturierung von Architekturmodellen entwickelt. Anschließend wurden domänenspezifische Architekturmodelle hinsichtlich ihrer Zwecke, dargestellter Elemente, Relationen sowie Visualisierung und Formalisierung analysiert. Die Analyse zeigt deutlich die Heterogenität und die domänenspezifische Ausprägung der Architekturmodelle. Aus der Analyse wurden grundlegende Anforderungen an die domänenübergreifende Architekturmodellierung abgeleitet. Die Ergebnisse und formulierten Anforderungen sind in die deskriptive Phase der Design Research Methodology [26] einzuordnen und stellen eine erste (unvollständige) Analyse dar. Die beschriebenen Ergebnisse müssen hinsichtlich ihrer Allgemeingültigkeit durch weiterführende Arbeiten abgesichert werden. Dieser Beitrag ist eher auf die Forschungscommunity limitiert, da zum jetzigen Zeitpunkt keine direkt anwendbare Vorgehensweise für die praktische Produktentwicklung abgeleitet wurden. Die formulierten Anforderungen sollen in Zukunft in Zusammenarbeit mit Industriepartner konkretisiert und erweitert werden. Zusätzlich soll ein grundsätzlicher Modellierungsansatz auf Basis des heterogenen Modellierungskonzepts ausgearbeitet und in verschiedenen Anwendungsfällen evaluiert werden. Schwerpunkt soll hierbei die Verknüpfung der unterschiedlichen Präsentationen in einem konsistenten Modell sein. Des Weiteren soll die Kopplung von formalen Modellierungen und weniger formalen Visualisierungen untersucht werden.

Danksagung

Der vorliegende Beitrag ist Teil der Arbeiten des niedersächsischen Zukunftslabors Mobilität. Das Teilprojekt wird gefördert vom Niedersächsischen Ministerium für Wissenschaft und Kultur (Fördernummer ZN3493) im Niedersächsischen Vorab der VolkswagenStiftung und betreut vom Zentrum für digitale Innovationen Niedersachsen (ZDIN).

Literaturverzeichnis

[1] VDI 2206: “Entwicklungsmethodik für mechatronische Systeme”, Beuth Verlag GmbH, 2004.

[2] Koppenhagen, F. “Modulare Produktarchitekturen – Komplexitätsmanagement in der frühen Phase der Produktentwicklung.” In: Schoeneberg KP. (eds) Komplexitätsmanagement in Unternehmen. Springer Gabler, Wiesbaden, 2014.

[3] Pahl, G., Beitz, W.: “Konstruktionslehre - Grundlagen erfolgreicher Produktentwicklung. Methoden und Anwendung”. Siebte Auflage. Berlin, Heidelberg, New York: Springer Verlag, 2007.

[4] Krause, D.; Vietor, T.; Inkermann, D.; Richter, T.; Hanna, M.; Wortmann, N.: “Produktarchitektur.” In: Bender, B.; Gericke, K. (Hrsg.): Pahl&Beitz Konstruktionslehre, Kapitel 13, Springer-Verlag, 2020 (in Veröffentlichung)

[5] Inkermann, D. et al.: “Die Produktarchitektur als zentrales Konzept in der Produktentwicklung”, In: Krause, D.; Wartzack, S.; Paetzold, K.: Tagungsband 30. DfX-Symposium, Yesteburg, 2019.

[6] Birkhofer, H.: “Analyse und Synthese der Funktionen technischer Produkte”, Düsseldorf, VDI-Verlag, 1980.

[7] Gausemeier, J., Dumitrescu, R., Steffen, D., Czaja, A., Wiederkehr, O., Tschirner, C.: “Systems Engineering in Industrial Practice”, Paderborn, Germany, 2015.

[8] Inkermann, D.: “Towards model-based process engineering”, International Conference of Engineering Design, ICED 19, Delft, 2019.

[9] Bertalanffy, L.v.: “General System Theory – Foundations, Development, Applications”. George Braziller, New York, 1969.

[10] Stachowiak, H.: “Allgemeine Modelltheorie” (German), Springer, Wien, 1973.

[11] Haberfellner, R., Weck, O. L. de, & Fricke, E.: “Systems Engineering: Fundamentals and applications”. Basel: Birkhäuser, 2019.

[12] Delligatti, L.: “SysML Distilled: A Brief Guide to the Systems Modeling Language”. 1st ed., Addison-Wesley Professional, 2013.

[13] Weilkiens, T.: **“**Systems Engineering with SysML/UML Modeling, Analysis, Design**”.** *Morgan Kaufmann OMG Press, 2008*

[14] Estefan, J.: “Survey of Model-Based Systems Engineering (MBSE) Methodologies” Jet Propulsion Laboratory California Institute of Technology, Pasadena, California, U.S.A., Rev. B, 2008.

[15] Rude, S.: “Wissensbasiertes Konstruieren”. Technische Hochschule Karlsruhe, Berichte aus dem Maschinenbau, 1998.

[16] Ponn, J.; Lindemann, U.: “Konzeptentwicklung und Gestaltung technischer Produkte. Systematisch von Anforderungen zu Konzepten und Gestaltlösungen”. 2. Aufl. Berlin – Heidelberg: Springer-Verlag, 2011.

[17] Kleiner, S.: „Entwerfen und Entwickeln mit Systems Engineering auf Basis des RFLP-Ansatzes in V6“, TUDpress – Verlag der Wissenschaft GmbH, Dresden, 2012.

[18] Dworschak, F. et al.: „Konzept für den MBSE-Einsatz zur automatisierten Individualisierung von komplexen Produkten“, In: Krause, D.; Wartzack, S.; Paetzold, K.: Tagungsband 30. DfX-Symposium, Jesteburg, 2019.

[19] Broy, M.; Damm, W. Henkler, S.; Pohl, K.; Vogelsang, A.; Weyer, T.: “Introduction to the SPES Modeling Fraework”. In: Pohl, K.; Hönninger, H.; Achatz, R.; Broy, M. (Eds.): Model-Based Engineering of Embedded Systems – The SPES 2020 Methodology. Berlin, Heidelberg: Springer-Verlag, 2012.

[20] VDI 2221: “Entwicklung technischer Produkte und Systeme, Blatt 1: Modell der Produktentwicklung”, Beuth, Düsseldorf, 2019.

[21] Wilms R. et al.: “Identifying Cross-Domain Linkage Types to Support Engineering Change Management and Requirements Engineering”, 29th CIRP Design, Elsevier B.V., 2019.

[22] Buur, J.; Andreasen, M.M.: “Design Models in Mechatronic Product Development.” In: Design Studies, Vol. 10, No. 3, pp. 155-162, 1989.

[23] Lunze, J.: “Regelungstechnik 1 - Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen”. Berlin, Heidelberg, New York: Springer-Verlag, 2005.

[24] Browning, T.R.: “Applying the design structure matrix to system decomposition and integration problems: a review and new directions”. In: IEEE Transactions on Engineering Management, Vol. 48, No. 3, pp. 292 – 306, 2001

[25] Jansen, S.: „Eine Methodik zur modellbasierten Partitionierung mechatronischer Systeme”. Dissertation, Universität Bochum, 2006.

[26] Blessing, L.T.M; Chakrabarti, A.: “DRM – A Design Research Methodology. Berlin, Heidelberg, New York: Springer-Verlag, 2009.