Band 420

Verlagsschriftenreihe des Heinz Nixdorf Instituts
Prof. Dr.-Ing. Roman Dumitrescu (Hrsg.)
Advanced Systems Engineering

Jörg Heihoff-Schwede

Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen

HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Jörg Heihoff-Schwede

Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen

Specification technique for the analysis, design and evaluation of engineering IT architectures

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen National-bibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Band 420 der Verlagsschriftenreihe des Heinz Nixdorf Instituts

© Heinz Nixdorf Institut, Universität Paderborn – Paderborn – 2024

ISSN (Online): 2365-4422
ISBN: 978-3-947647-39-2

Das Werk einschließlich seiner Teile ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung der Herausgeber und des Verfassers unzulässig und strafbar. Das gilt insbesondere für Vervielfältigung, Übersetzungen, Mikroverfilmungen, sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Als elektronische Version frei verfügbar über die Digitalen Sammlungen der Universitätsbibliothek Paderborn.

Satz und Gestaltung: Jörg Heihoff-Schwede

Geleitwort

Das Model-based Systems Engineering (MBSE) zur modellbasierten Realisierung komplexer technischer Systeme stellt einen Kern der Forschung des Lehrstuhls Advanced Systems Engineering am Heinz Nixdorf Institut und dem damit verbundenen Fraunhofer-Institut für Entwurfstechnik Mechatronik IEM in Paderborn dar.

Unternehmen stehen aufgrund der Digitalisierung und dem sich veränderndem Wettbewerbsumfeld vor einem Wandel. Ein wesentliches Element dabei ist der Wandel zu autonomen, interaktiven und dynamischen Produkten. Advanced Systems Engineering (ASE) stellt hier einen Handlungsrahmen zur Begegnung der resultierenden Herausforderungen dar. Für die Unternehmen gilt es jedoch, die daraus resultierenden neuen Prozesse, Methoden und digitalen Werkzeuge in ihr Engineering zu integrieren. Dabei sind eine Vielzahl von Wechselwirkungen zwischen den Prozessen, Methoden und den verschiedenen Werkzeugen zu berücksichtigen.

Vor diesem Hintergrund hat Herr Heihoff-Schwede eine Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen erarbeitet. Er unterstützt somit die modellbasierte Analyse der Prozess- und Methodenelemente sowie der daraus resultierenden Daten und darauf basierend die Ableitung und Bewertung geeigneter Architekturen von IT-Werkzeugen. Den Kern der Arbeit bilden ein Vorgehensmodell, ein Sprachkonzept, welches eine geeignete Modellierungssprache und Hilfsmittel umfasst, sowie eine Werkzeugunterstützung. Die Validierung erfolgte bei einem Industrieunternehmen im Rahmen der Einführung von MBSE.

Mit seiner Arbeit hat Herr Heihoff-Schwede einen wertvollen Beitrag zur praktischen Umsetzung von ASE im industriellen Umfeld geleistet. Die Spezifikationstechnik leistet einen praxisorientierten Beitrag im Feld des Enterprise Architecture Managements (EAM). Damit trifft die Arbeit den Nerv der Zeit, da sich das Thema EAM aktuell im Forschungsfeld des ASE im Heinz Nixdorf Insitut und des Fraunhofer IEMs hoher Relevanz erfreut. Ich wünsche Herrn Heihoff-Schwede weiterhin viel Erfolg bei der Einführung von ASE und MBSE in seinem Arbeitsumfeld.

Paderborn, im April 2024 *Prof. Dr.-Ing. Roman Dumitrescu*

Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen

zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät Maschinenbau
der Universität Paderborn

genehmigte
DISSERTATION

von
M.Sc. Jörg Heihoff-Schwede
aus *Salzkotten*

Tag des Kolloquiums:	18.12.2023
Referent:	Prof. Dr.-Ing. Roman Dumitrescu
Korreferent:	Prof. Dr.-Ing. Jürgen Gausemeier
Korreferentin:	Prof. Dr.-Ing. Lydia Kaiser

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Entwurfstechnik Mechatronik IEM. Sie ist das Ergebnis meiner wissenschaftlichen Arbeit in Industrieprojekten.

Mein herzlicher Dank gilt Prof. Dr.-Ing. Roman Dumistrescu, der mich als Student mit dem Thema Systems Engineering in Kontakt gebracht hat und mir die Möglichkeit gegeben hat, mich in diesem Feld anhand meiner persönlichen Interessen und Begabungen weiterzuentwickeln. Vielen Dank für deine Offenheit, dein Vertrauen und deine Unterstützung auf dem Weg der Dissertation.

Ich danke Prof. Dr.-Ing. Jürgen Gausemeier, Senior Professor des Fachgebietes Strategische Produktplanung und Systems Engineering am Heinz Nixdorf Institut der Universität Paderborn sowie Prof. Dr.-Ing. Lydia Kaiser, ECDF-Professorin für Digitales Engineering 4.0 an der Technischen Universität Berlin für die Übernahme des Korreferates.

Danken möchte ich allen Kolleginnen und Kollegen am Fraunhofer IEM für die tolle und kollegiale Zusammenarbeit. Die Zeit mit euch wird mir immer in Erinnerung bleiben! Besonders möchte ich mich bei Prof. Dr.-Ing. Lydia Kaiser und Dr.-Ing. Christian Tschirner für die Unterstützung bei der Themenfindung und euerem Feedback zur Dissertation bedanken. Hervorheben möchte ich auch Dr.-Ing. Lukas Bretz und M.Sc. Russlan Bernijazov, die mich durch spannende Diskussionen im Model-based Systems Engineering unterstützt und damit auch meine Dissertation bereichert haben. Darüber hinaus danke ich den Studierenden am Fraunhofer IEM, insbesondere Lara Meyer für die Unterstützung bei der LaTeX-Vorlage und Nura Yanik sowie Simon Erbertz für die Hilfe bei der Gestaltung von Grafiken.

Erwähnen möchte ich an dieser Stelle die Kollegen beim langjährigen Industriepartner CLAAS, die es mir ermöglicht haben, meine Arbeit an den praktischen Belangen auszurichten und zu validieren. Mein besonderer Dank gilt Dr.-Ing. Kai Korthals, Nico Michels und Jan Gröver.

Abschließend möchte ich meinen Freunden und meiner Familie danken. Vielen Dank an Hildegard und Jan für euer Korrektorat dieser Arbeit. Meinen Eltern Meinolf und Brigitte, weil ihr mir diesen Weg überhaupt erst ermöglicht und mich immer bei meinem Werdegang unterstützt habt. Danke an meinen Freund Julian, der mich während der Erstellung der Dissertation begleitet und dabei viel Geduld und Verständnis aufgebracht hat.

Paderborn, im April 2024 *Jörg Heihoff-Schwede*

Zusammenfassung

Technische Systeme und ihr Umfeld verändern sich grundlegend und ihre Komplexität steigt. Das macht es notwendig, das Engineering dieser Systeme daran anzupassen. Ansätze wie das Model-based Systems Engineering setzen in vielen Unternehmen umfassende Änderungen bei den Prozessen, Methoden, Modellen und IT-Werkzeugen im Engineering voraus. Die Prozesse, Methoden und Modelle müssen dazu häufig zusammenhängend betrachtet werden, um daraus konsistente Anforderungen an die IT-Architektur abzuleiten. Für diese ergibt sich oftmals ein komplexer Lösungsraum mit einer Vielzahl von IT-Werkzeugen und Schnittstellen, wobei die Anforderungserfüllung der verschiedenen Lösungsvarianten sehr unterschiedlich sein kann. Dies erschwert aufgrund vieler Abhängigkeiten nicht nur die Bildung der Lösungsvarianten, sondern auch deren Bewertung. Ganzheitliche und zusammenhängende Modelle der Prozesse, Methoden, Modelle und IT-Werkzeuge des Engineerings können helfen, diese Aufgaben systematisch und effektiv zu unterstützen.

Ziel dieser Arbeit ist eine Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen. Im Kern besteht diese aus einem Sprachkonzept, welches Sichten, eine Modellierungssprache und die Vernetzung der Modellelemente definiert. Ein Vorgehensmodell leitet den Anwender an und erlaubt in Kombination mit einer Werkzeugunterstützung die praktische Anwendung. Ein industrielles Beispiel aus der Landtechnik demonstriert den Einsatz der Spezifikationstechnik.

Summary

Technical systems and their environment are changing fundamentally and their complexity is increasing. This makes it necessary to adapt the engineering of these systems to this situation. In many enterprises, approaches such as model-based systems engineering require extensive changes in processes, methods, models and IT tools in engineering. The processes, methods and models often have to be considered as a whole in order to derive consistent requirements for the IT architecture. These often result in a complex solution space with a variety of IT tools and interfaces, whereas the fulfilment of requirements of the different solution variants can differ widely. Due to many dependencies, this not only makes it difficult to form the solution variants, but also to evaluate them. Holistic and coherent models of the processes, methods, models and IT tools of engineering can help to support these tasks systematically and effectively.

The goal of this thesis is a specification technique for the analysis, design and evaluation of engineering IT architectures. In essence, it consists of a language concept, which defines views, a modelling language and the interconnection of the model elements. A process model guides the user and, in combination with a tool support, allows the practical application. An industrial example from agricultural engineering demonstrates the use of the specification technique.

Liste der veröffentlichten Teilergebnisse

[HBK+19] HEIHOFF-SCHWEDE, J.; BRETZ, L.; KAISER, L.; DUMITRESCU, R.: An Explanatory Model as Basis for the Introduction of Systems Engineering and capable IT-Infrastructures in Industry. In: Proceedings of the 5th Annual IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, Scottland, UK, October 1-3, 2019, 2019

[HBR+16] HEIHOFF-SCHWEDE, J.; BREMER, C.; RABE, M.; TSCHIRNER, C.: Werkzeuge für den Mittelstand - MBSE leicht. In: Schulze, S.-O.; Tschirner, C.; Kaffenberger, R. (Hrsg.), Tag des Systems Engineering, Herzogenaurach, 25-27. Oktober 2016, 2016, S. 35–44

[HKD19a] HEIHOFF-SCHWEDE, J.; KAISER, L.; DUMITRESCU, R.: An MBSE-based approach for the analysis of requirements towards engineering IT architectures. In: Proceedings of the 5th Annual IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, Scottland, UK, October 1-3, 2019, 2019

[HKD19b] HEIHOFF-SCHWEDE, J.; KAISER, L.; DUMITRESCU, R.: An MBSE-based approach for the functional analysis of engineering IT architectures. In: Schulze, S.-O.; Tschirner, C.; Kaffenberger, R. (Hrsg.), Tag des Systems Engineering, München, 6.-8. November 2019, Gesellschaft für Systems Engineering, 2019

[HKD20] HEIHOFF-SCHWEDE, J.; KAISER, L.; DUMITRESCU, R.: An MBSE-based approach for the definition and evaluation of engineering IT architectures. In: Marjanovic, D.; Storga, M.; Pavkovic, N.; Bojcetic, N. (Eds.), DS 102 – Proceedings of the DESIGN 2020 16th International Design Conference, October 26-29, 2020, Cambridge University Press, 2020, pp. 2285–2294

[MIS+18] MIROSHNIKOV, S.; IWANEK, P.; SCHUMACHER, M.; HEIHOFF-SCHWEDE, J.; GREINERT, M.: Conceptual Design and Analysis of Mechatronic Systems based on Function Catalogues. In: Tag des Systems Engineerings & INCOSE EMEA Sector Systems Engineering Conference 2018, Berlin, Germany, November 5-7, 2018, 2018

Inhaltsverzeichnis

Seite

1 Einleitung 5
1.1 Problematik 5
1.2 Zielsetzung 7
1.3 Aufbau dieser Arbeit 8

2 Forschungsdesign 10
2.1 Ansatz der Design Research Methodology 10
2.2 Ansatz der Konsortialforschung 12
2.3 Einordnung und Vorgehen in dieser Arbeit 14

3 Problemanalyse 16
3.1 Begriffsdefinitionen und Einordnung der Arbeit 16
3.2 Evolution Technischer Systeme 20
3.3 Engineering technischer Systeme 23
3.3.1 Prozesse 23
3.3.2 Methoden 26
3.3.3 IT-Werkzeuge 27
3.3.4 Artefakte 27
3.3.5 Organisationsstrukturen und Kompetenzen 28
3.3.6 Gestaltung des Engineerings 28
3.4 Engineering-IT 31
3.4.1 Aufgaben der Engineering-IT 32
3.4.2 Rollen in der Engineering-IT 32
3.4.3 IT-Werkzeuge im Engineering 34
3.4.4 Anpassungsmöglichkeiten von IT-Werkzeugen im Engineering . . 39
3.4.5 Integration der IT-Werkzeuge im Engineering 41
3.4.6 Treiber zur Veränderung der Engineering-IT 44
3.4.7 Systemische Betrachtung der Engineering-IT 49
3.4.8 Gestaltung der Engineering-IT-Infrastruktur 51
3.5 Problemabgrenzung 55
3.6 Anforderungen 59

4 Stand der Technik 61
4.1 Ganzheitliche Ansätze zur Architekturbeschreibung 61
4.1.1 Beschreibung von Architekturen nach ISO/IEC/IEEE 42010 61
4.1.2 Sprachen zur Beschreibung von Architekturen 63
4.1.3 Sprachen zur Beschreibung von Prozessen und Methoden 69
4.1.4 Spezifikation von Wertschöpfungssystemen nach SCHNEIDER . . 76

4.2 Ansätze zur Analyse von Anforderungen und Prozessen 78
4.2.1 Ansätze zum Requirements Engineering Technischer Systeme . . 78
4.2.2 Geschäftsprozessmanagement nach SCHMELZER UND SESSELMANN 81
4.2.3 Model-Based Process Engineering nach HUTH ET AL. 82
4.3 Ansätze zur Definition von Enterprise und IT-Architekturen 84
4.3.1 The Open Group Architecture Framework (TOGAF) 84
4.3.2 Informationssystem-Architekturen nach HOFFMANN UND HEIMES . 87
4.3.3 Modellbasierte Spezifikation einer MBSE-Modellierungsumgebung nach FRIEDLAND ET AL. 89
4.3.4 Funktionale Architekturen für Systeme nach LAMM UND WEILKIENS 91
4.4 Ansätze zur Auswahl von IT-Werkzeugen 93
4.4.1 Nutzwertbasierte Evaluationsansätze für Standardsoftware 93
4.4.2 Alternative Evaluationsansätze für Standardsoftware 97
4.5 Handlungsbedarf 99

5 Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen 103
5.1 Die Spezifikationstechnik im Überblick 104
5.2 Vorgehensmodell zur Spezifikation der Engineering-IT 105
5.2.1 Phase 1: Modellierung der Ausgangssituation 106
5.2.2 Phase 2: Spezifikation des Zielbildes 108
5.2.3 Phase 3: Definition des Zielsystems 109
5.2.4 Phase 4: Definition und Bewertung von Lösungsvarianten 110
5.2.5 Iterativer Durchlauf des Vorgehensmodells 114
5.3 Sprachkonzept 114
5.3.1 Modellierungskonzept 115
5.3.1.1 Metamodell 115
5.3.1.2 Grundlagen des Modellierungskonzepts 117
5.3.1.3 Sichtenkonzept 118
5.3.1.4 Vernetzung der Partialmodelle 121
5.3.2 Sprachkonstrukte und ihre Anwendung 123
5.3.2.1 Übergreifende Sprachkonstrukte 123
5.3.2.2 Prozessmodell 125
5.3.2.3 Methodenmodell 130
5.3.2.4 Anwendungsfälle 132
5.3.2.5 Funktionale Architektur 133
5.3.2.6 Anforderungen 136
5.3.2.7 Lösungsmodell 138
5.3.2.8 Komponentenarchitektur 145
5.4 Werkzeugunterstützung 148
5.4.1 Kartenset zur Workshopmodellierung 148
5.4.2 Sprachprofil zur rechnergestützten Modellierung 149

5.4.3 Softwarewerkzeug zur rechnergestützten Modellierung 150

6 Anwendung und Bewertung 153
6.1 Anwendung der Spezifikationstechnik am Beispiel 153
6.1.1 Einführung des Anwendungsbeispiels 153
6.1.2 Phase 1: Modellierung der Ausgangssituation 154
6.1.3 Phase 2: Spezifikation des Zielbildes 157
6.1.4 Phase 3: Definition des Zielsystems 158
6.1.5 Phase 4: Definition und Bewertung von Lösungsvarianten 162
6.2 Bewertung der Arbeit an den Anforderungen 166

7 Zusammenfassung, Reflexion und Ausblick 169

Abkürzungsverzeichnis 173

Literaturverzeichnis 175

Anhang

A1 Ergänzungen zur Spezifikationstechnik A-1
A1.1 Festlegungen zur grafischen Notation A-1
A1.2 Zuordnung der Partialmodelle zu Viewpoints A-2
A1.3 Profil der Spezifikationstechnik A-3

A2 Beschreibung der Viewpoints A-5
A2.1 Viewpoints der Prozess Domäne A-5
A2.2 Viewpoints der Methoden Domäne A-6
A2.3 Viewpoints der Funktions Domäne A-7
A2.4 Viewpoints der Applikations Domäne A-8
A2.5 Viewpoints der Anforderungs Domäne A-10

A3 Methodische Hilfsmittel A-13
A3.1 Paarweiser Vergleich mittels AHP A-13
A3.2 Bewertung der User Experience anhand des UEQ A-15

A4 Übersetzungen aus dem Englischen A-18

1 Einleitung

Die vorliegende Arbeit ist im Rahmen der anwendungsorientierten Forschung am Fraunhofer-Institut für Entwurfstechnik Mechatronik IEM entstanden. Die Ergebnisse resultieren aus verschiedenen Forschungs- und Industrieprojekten. Stellvertretend seien an dieser Stelle zwei Industrieprojekte mit einem Unternehmen aus der Landtechnik zu nennen. Das erste Projekt hatte das Ziel, Anforderungen eines Requirement Engineering-IT-Werkzeugs für eine neue, in das Unternehmen zu integrierende Methode zu identifizieren. Dazu wurde die Methode systematisch hinsichtlich relevanter Aktivitäten und Artefakte untersucht und daraus Anforderungen und Anwendungsfälle abgeleitet. Dieser Aspekt ist als ein wesentliches Element in die Arbeit eingeflossen und legt die Grundlage für die Modelle in Kapitel 5.3.2. Ein weiteres Projekt hatte das Ziel, das Engineering als System mit seinen Schnittstellen abzugrenzen, wesentliche Prozesse und Artefakte zu modellieren und diese mit geeigneten Merkmalen zur Ableitung von Anforderungen an die Engineering-IT zu verknüpfen. Dazu wurde ein Prozessmodell erstellt, ein geeignetes Metamodell definiert und ein dazugehöriges Modellierungskonzept entwickelt. Das Projekt hat dadurch einen maßgeblichen Beitrag zum Modellierungskonzept in Kapitel 5.3.1 geleistet. Der Autor dieser Arbeit ist zudem im Anschluss an die Tätigkeit als wissenschaftlicher Mitarbeiter bei dem zuvor genannten Unternehmen tätig und verantwortet dort die Prozesse, Methoden, Datenmodelle und IT-Werkzeuge für das Requirements Engineering. Auch im Rahmen dieser Tätigkeit wurde die vorliegende Arbeit angewandt und weiter verfeinert.

In den folgenden Kapiteln 1.1 und 1.2 werden die Problematik sowie die Zielsetzung der vorliegenden Arbeit dargelegt. Kapitel 1.3 beschreibt den Aufbau dieser Arbeit.

1.1 Problematik

Technische Erzeugnisse und ihr Umfeld verändern sich in der heutigen Zeit deutlich. So haben sich rein mechanische und mechatronische Systeme hin zu **intelligenten Technischen Systemen** entwickelt. Sie können adaptiv auf ihr Umfeld reagieren, zeigen ein robustes Verhalten mit autonomen Reaktion auf dynamische Veränderungen, versprechen eine höhere Effektivität durch proaktives Verhalten zur Zieloptimierung und eine höhere Benutzerfreundlichkeit, indem sie individuelles Benutzerverhalten berücksichtigen [Dum10, S. 41]. Diese auch als Cyber Physical Systems bezeichneten Systeme besitzen somit eine inhärente Intelligenz und bilden die Grundlage für smarte Anwendungen, z. B. Smart Products, Smart Logistics oder Smart Factories. Dies wird besonders im Bereich der Produktion deutlich. Die Digitalisierung und Vernetzung von Maschinen, Produkten und angrenzenden Systemen sowie Abläufen zu effizienten Wertschöpfungssystemen hat hier den Begriff Industrie 4.0 geprägt [GDE+18, S. 86f].

Diese eindrucksvolle Entwicklung geht jedoch mit einer höheren Komplexität der Systeme einher. Daraus resultieren verschiedene **Herausforderungen in der Entwicklung der**

technischen Systeme. Hier ist eine wachsende Interdisziplinarität zu beobachten sowie eine Vielzahl heterogener Schnittstellen und Wechselwirkungen zu berücksichtigen. Die Komplexität der Anforderungen und des Funktionsumfangs der Systeme steigen, gleichzeitig nimmt die Produktvarianz zu [GDS+13, S. 16]. Es ist daher unvermeidbar, dass sich die steigende Komplexität der technischen Systeme auch in ihrer Entwicklung, dem Engineering, niederschlägt. Das Engineering muss dementsprechend auf die Entwicklung von intelligenten technischen Systemen angepasst werden [TG18, S. 7].

Als Reaktion darauf entwickeln sich Ansätze wie das Smart Engineering [AES+12, S. 10ff] oder Advanced Systems Engineering (ASE) [DAR+21, S. 28ff]. **Gestaltungsoptionen zur Anpassung des Engineerings** stellen insbesondere die Prozesse, Methoden, Artefakte und IT-Werkzeuge dar [Abr18, S. 5ff]. Betrachtet man die aktuelle Entwicklung, so lassen sich zwei wesentliche **Treiber für die Komplexität im Engineering** finden. Erstens nimmt die Anzahl und Vielfalt der ingenieurwissenschaftlichen Ansätze und der daraus resultierenden Artefakte[1] zu [AZL19, S. 1]. Neue Prozesse, die beschreiben, *was* zu tun ist, und Methoden, die beschreiben, *wie* etwas zu tun ist, müssen in Unternehmen implementiert werden. Betrachtet man beispielsweise das ASE, so setzt dieses die Implementierung einer Vielzahl von Prozessen und Methoden voraus. Da ASE ein interdisziplinärer Entwicklungsansatz ist, führt er zwangsläufig zu disziplin- und domänenübergreifenden Artefakten. Diese Artefakte müssen durch übergreifende und konsistente Lebenszyklusmanagement-Prozesse, insbesondere dem Konfigurationsmanagement, aktiv gesteuert werden [ISO15288, S. 39f]. Zudem wird aufgrund normativer [DIN61508, S. 52ff] oder methodischer Forderungen [FBK+14, S. 23ff] immer häufiger die Rückverfolgbarkeit zwischen Artefakten gefordert, was oftmals werkzeugübergreifende Verknüpfungen voraussetzt [DAR+21, S. 65]. Zweitens wirkt sich auch die Digitalisierung auf das Engineering aus. Artefakte werden digitalisiert und müssen über definierte und formalisierte Datenstrukturen abgebildet werden, wie bspw. durch das Model-based Systems Engineering deutlich wird. Darüber hinaus impliziert die digitale Transformation eine Durchgängigkeit der Daten zur Unterstützung von End-to-End Prozessen und ihre Automatisiertung, wie es beispielsweise für das Konzept des digitalen Zwillings notwendig ist [BHS15, S. 70]. Das wirkt sich insbesondere auf die Engineering-IT aus, also die Informationstechnik (speziell die IT-Werkzeuge), welche zur lebenszyklusübergreifenden Produkt- und Serviceentwicklung notwendig ist.

Dies führt zu neuen **Herausforderungen in der Gestaltung der Engineering-IT**, da die Komplexitätstreiber neue Anforderungen aufwerfen, die durch die IT-Werkzeuge erfüllt werden müssen. IT-Werkzeuge erfüllen nur einen begrenzten Funktionsumfang, weshalb aufgrund der zunehmenden Vielfalt der Prozesse und Methoden auch eine Vielzahl von IT-Werkzeugen notwendig ist [AZL19, S. 1ff], [FNF+14, S. 211]. Die Prozesse und Methoden nutzen verknüpfte Daten, weshalb auch die Verbindung der IT-Werkzeuge untereinander an Relevanz gewinnt. Ihre Schnittstellen sind jedoch häufig nicht kompatibel oder

[1] Ein Artefakt ist im Sinne dieser Arbeit eine Information, die im Lebenszyklus eines Systems verwendet oder erzeugt wird (z. B. im Rahmen der Entwicklungstätigkeit oder dessen Betrieb) [OMG17-ol, S. 656], [Rop09, S. 146].

erlauben nur einen eingeschränkten Funktionsumfang, was eine ganzheitliche Planung der IT-Architekturen notwendig macht [Lu19, S. 3ff]. Zudem weist jedes IT-Werkzeug unterschiedliche Eigenschaften auf, insbesondere bzgl. der Funktionalität und Gebrauchstauglichkeit. Dies führt zu der Fragestellung, welche Kombination von IT-Werkzeugen die Anforderungen am besten erfüllt [ISO25010, S. 26ff]. In der Praxis kann dies oft ein Dilemma darstellen, da Werkzeuge, die ausschließlich nach hoher Funktionalität oder Gebrauchstauglichkeit ausgewählt werden, zu föderierten IT-Architekturen mit komplexen Schnittstellen führen. Dies kann die Durchgängigkeit einschränken und zu hohen Implementierungs- und Wartungsaufwänden führen. Die bisher übliche Ad-hoc Planung von Werkzeugketten ist daher nicht länger zielführend [AZL19, S. 9]. Vielmehr ist ein Ansatz im Sinne des Business Engineerings erforderlich, d.h. ein ganzheitliches, über mehrere Gestaltungsebenen gegliedertes Transformationsmanagement, welches sich eines ingenieurmäßigen Vorgehens bedient [Lei15, S. 12f]. Da das Business Engineering und vergleichbare Ansätze wie das Enterprise Architecture Management (s. Kap. 3.4.8 und 4.3) jedoch nicht die Besonderheiten im Engineering berücksichtigen, soll hier der Begriff der Engineering-IT-Architektur geprägt werden. Diese umfasst, angelehnt an die Definition des Architektur-Begriffs, die grundsätzlichen Konzepte und Strukturen der Prozesse, Methoden, Artefakte und IT-Werkzeuge im Engineering sowie ihre Beziehungen untereinander [ISO42010, S. 2ff]. Es ist somit ein **ganzheitlicher und systematischer Ansatz** zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen notwendig.

Ein weiteres Problemfeld liegt in der Kommunikation und Dokumentation der Ergebnisse aus Analyse- und Entwurfstätigkeiten der Engineering-IT-Architektur. Modelle stellen hier ein geeignetes Hilfsmittel zur Beschreibung dar [TOG18, S. 6ff]. Bei der Entwicklung einer Engineering-IT-Architektur sind zahlreiche Stakeholder mit verschiedenen Hintergründen beteiligt, z. B. Prozessverantwortliche oder IT-Architekten [Lu19, S. 6]. Daher besteht die Herausforderung, die Modellinhalte für alle Stakeholdergruppen verständlich darzustellen. Da das Modell zur Dokumentation, Spezifikation und Kommunikation dient, sind eindeutige Sprachkonstrukte sowie Regeln zur Dokumentation des Analyse- und Designprozesses sinnvoll, um Eindeutigkeit und Standardisierung zu gewährleisten [BRS95, S. 439ff]. Darüber hinaus ist häufig keine Rückverfolgbarkeit gegeben, um die Auswirkungen von Änderungen und Entscheidungen nachvollziehen zu können. Das Modell einer Engineering-IT-Architektur kann im Rahmen dieser Arbeit also als Beschreibung der Engineering-IT-Architektur definiert werden, also ein ganzheitliches Abbild der Prozesse, Methoden, Artefakte und IT-Werkzeuge im Engineering und ihr Zusammenspiel.

1.2 Zielsetzung

Ziel der vorliegenden Arbeit ist eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*. Sie soll Dritte befähigen, mittels modellbasierter Beschreibungstechniken die Prozesse, Methoden und Artefakte zusammenhängend zu dokumentieren, analysieren und zu optimieren. Darüber hinaus unterstützt sie, für das resultierende Zielbild geeignete Architekturen der IT-Werkzeuge abzuleiten, diese ganz-

heitlich zu bewerten und schließlich die bestmögliche Kombination von IT-Werkzeugen auszuwählen. Das Resultat stellt ein Modell der Engineering-IT-Architektur dar, welches die Prozesse, Methoden, Artefakte und IT-Werkzeuge im Unternehmen zusammenhängend dokumentiert.

Die Spezifikationstechnik soll sich aus drei Bestandteilen zusammensetzen: einem Vorgehensmodell, einem Sprachkonzept und einer Werkzeugunterstützung. Das **Vorgehensmodell** befähigt die Anwender zur systematischen Erstellung einer Engineering-IT-Architektur. Es beschreibt dazu notwendige Phasen und Tätigkeiten zur Anwendung des Sprachkonzepts und dazu geeignete Hilfsmittel. Das **Sprachkonzept** definiert eine Modellierungssprache mit entsprechenden Sprachkonstrukten zur zusammenhängenden Beschreibung von Prozessen, Methoden, Artefakten und Anwendungsarchitekturen im Engineering. Es legt zudem grundsätzliche Konzepte zur Arbeit mit dem Modell der Engineering-IT-Architektur fest, indem es die systematische Bildung von bedarfsgerechten Ausschnitten auf das Modell (Sichten) und die Vernetzung der Modellelemente definiert. Die **Werkzeugunterstützung** beinhaltet ergänzende Hilfsmittel zur Anwendung des Sprachkonzepts.

1.3 Aufbau dieser Arbeit

Die vorliegende Arbeit gliedert sich in sieben Kapitel. Anschließend an die Einleitung (Kapitel 1) wird in **Kapitel 2** das Forschungsdesign dieser Arbeit erläutert. Dazu werden die Änsätze der Design Research Methodology und der Konsortialforschung erläutert. Basierend darauf wird das Vorgehen zur Entwicklung dieser Arbeit beschrieben.

In **Kapitel 3** wird die bereits skizzierte Problemstellung vertieft. Zunächst werden die Evolution der technischen Systeme hin zu intelligenten technischen Systemen und die damit verbundenen Herausforderungen aufgezeigt. Nachfolgend werden mit Prozessen, Methoden, Artefakten und IT-Werkzeugen die Gestaltungsoptionen zur Anpassung des Engineerings an die Herausforderungen dargelegt. Das Engineering umfasst im Rahmen dieser Arbeit die Produktentwicklung sowie die Schnittstellen zu angrenzenden Bereichen. Vor diesem Hintergrund werden die Aufgaben und Rollen der Engineering-IT als Organisationsbereich sowie typische IT-Werkzeuge im Engineering und ihre Anpassung bzw. Integration in eine unternehmensspezifische IT-Architektur vorgestellt. Es folgt eine Analyse der Treiber zur Veränderung der Engineering-IT. Die Theorie des Systems Engineering und des Enterprise Architekturmanagement werden als Ansätze zur systematischen Gestaltung der Engineering-IT-Architektur vorgestellt. Im Rahmen der Problemabgrenzung werden die Nutzenpotentiale und Herausforderungen zusammengefasst und der Bedarf definiert. Schließlich werden Anforderungen an die zu entwickelnde Spezifikationstechnik abgeleitet.

Darauf aufbauend adressiert **Kapitel 4** den Stand der Technik. In diesem Zuge werden ganzheitliche Ansätze zur Architekturbeschreibung untersucht. Nachfolgend werden Ansätze zur Analyse von Anforderungen und Prozessen betrachtet. Es folgen Ansätze zur Definition von Enterprise und IT-Architekturen sowie zur Auswahl von IT-Werkzeugen.

Abschließend werden die Ansätze im Hinblick auf die gestellten Anforderungen bewertet und der Handlungsbedarf abgeleitet.

Kapitel 5 beschreibt die *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*. Einleitend wird ein Überblick über die Bestandteile der Spezifikationstechnik gegeben. Zunächst wird das Vorgehensmodell beschrieben, welches Phasen, Tätigkeiten und Hilfsmittel zur Erstellung einer Engineering-IT-Architektur definiert und so den Anwender anleitet. Nachfolgend wird das Sprachkonzept beschrieben. Ein Metamodell der Modellierungssprache definiert die Sprachelemente und ihre Zusammenhänge. Zudem werden grundlegende Konzepte zur systematischen Bildung von zweckspezifischen Ausschnitten auf das Modell sowie die Vernetzung des Modells erläutert. Nachfolgend werden die Sprachkonstrukte und ihre Semantik im Detail beschrieben. Abschließend wird die Werkzeugunterstützung in Form eines Kartensets für Workshops vorgestellt. Dieses wird durch ein kommerzielles Modellierungswerkzeug ergänzt, für welches ein formalisiertes Sprachprofil und ein Plug-in zur Anpassung der Benutzeroberfläche bereitgestellt werden.

Kapitel 6 beschreibt die Anwendung im Rahmen eines durchgängigen Beispiels. Dazu wird zunächst in das praktische Anwendungsbeispiel aus der Landtechnik eingeführt. Zur Bewertung der Praxistauglichkeit wird die Spezifikationstechnik auf das Anwendungsbeispiel angewandt. Abschließend wird die Spezifikationstechnik hinsichtlich der Erfüllung der an sie gestellten Anforderungen bewertet.

In **Kapitel 7** werden die Inhalte der Arbeit zusammengefasst und reflektiert. Es wird ein Ausblick auf zukünftigen Forschungsbedarf für die Analyse, Gestaltung und Evaluierung von Engineering-IT-Architekturen gegeben.

2 Forschungsdesign

Dieses Kapitel beschreibt das Forschungsdesign der vorliegenden Arbeit. Es ordnet dazu in Kap. 2.1 das Forschungsfeld Design Research ein und zeigt mit der Design Research Methodology einen etablierten Ansatz zur Lösung von Forschungsfragen in diesem Bereich auf. Kapitel 2.2 stellt aufgrund der engen Einbindung des Praxisunternehmens den Ansatz der Konsortialforschung vor. In Kapitel 2.3 wird dann das konkrete Vorgehen dieser Arbeit vorgestellt.

2.1 Ansatz der Design Research Methodology

Als Design wird durch BLESSING UND CHAKRABARTI der Prozess zum Identifizieren von Bedürfnissen und die Entwicklung einer zugehörigen Lösung, eines Produkts, bezeichnet. Das erzeugte Produkt soll die Bedürfnisse seiner Nutzer oder Stakeholder befriedigen [BC09, S. 1, 12]. Nach WYNN UND CLARKSON kann Design auch auf den Produktentstehungsprozess bezogen werden, das Produkt im Kontext des Designs kann also das Engineering darstellen [WC18, S. 162]. Die vorliegende Arbeit fällt somit in den Bereich des Design Research. Design Research ist die Generierung von Wissen über und für Design, mit dem Ziel, die Anwendung von Design durchzuführen [Hor01, S. 13], [BC09, S. 12].

BLESSING UND CHAKRABARTI stellen mit der Design Research Methodology (DRM) einen domänenübergreifenden Ansatz zur zielgerichteten Planung und Durchführung von Design Research vor. Das Vorgehen ist in Bild 2-1 dargestellt und umfasst vier Phasen: Klärung der Forschungsfrage, deskriptive Studie 1, präskriptive Studie und deskriptive Studie 2 [BC09, S. 12ff].

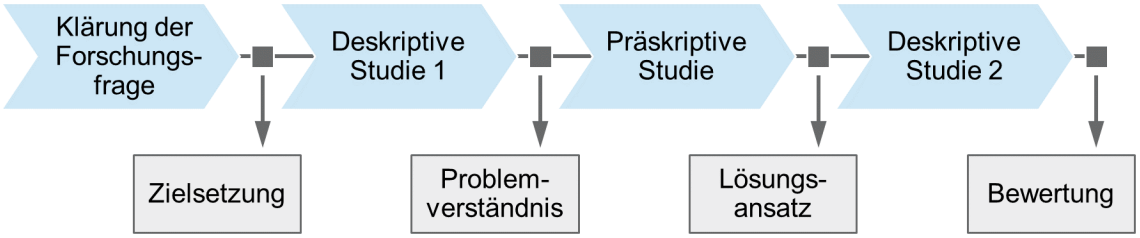


Bild 2-1: Vorgehen der Design Research Methodology nach [BC09, S. 15] und [Jop21, S. 11]

Die erste Phase umfasst die **Klärung der Forschungsfrage**. Sie zielt darauf ab, Ziele für die Forschungsarbeit zu definieren, die in deren Rahmen umgesetzt werden sollen. Dazu sind Abhängigkeiten zu untersuchen, die auf das Forschungsziel einwirken. Dies kann beispielsweise ein Abgleich zwischen der vorherrschenden und gewünschten Situation sein. Es gilt messbare Erfolgskriterien zu definieren, gegen die in Phase vier bewertet werden kann [BC09, S. 29ff, 43ff].

Die zweite Phase dient der Durchführung einer **ersten deskriptiven Studie**. Sie hat zum Ziel, das Verständnis zur Problemstellung bzw. Forschungsfrage zu vertiefen. Von besonderer Relevanz ist hier, die Probleme mit Erfolgsfaktoren zu verknüpfen, um einen Beitrag zur Problemlösung leisten zu können. Dazu können insbesondere Literaturrecherchen, empirische Untersuchungen, aber auch argumentative Beweisführungen zum Einsatz kommen [BC09, S. 31ff, 75ff].

In der dritten Phase wird eine **präskriptive Studie** durchführt. Diese setzt auf dem zuvor erarbeiteten Problemverständnis und somit auf den Ergebnissen der zweiten bzw. vierten Phasen auf. Ziel der Phase ist die Entwicklung eines konkreten Lösungsansatzes. Für dessen Entwicklung sollten die vielversprechendsten Erfolgsfaktoren ausgewählt werden und der Lösungsansatz diese systematisch adressieren. Dabei sind bereits existierende Lösungen zu beachten und schon die Evaluierung zu berücksichtigen [BC09, S. 33ff, S. 141ff]. Bei der Entwicklung des Lösungsansatzes bieten sich allgemeine Problemlösungs- und Entwicklungstechniken an, wie bspw. funktionale Analysen, morphologischer Kasten oder Brainstorming [BC09, S. 291ff].

Die vierte Phase umfasst eine **zweite deskriptive Studie**. Ziel ist die Bewertung des Lösungsansatzes. Dazu kommen empirische Ansätze zum Einsatz, um die Anwendbarkeit und den Erfolg des Lösungsansatzes zu erheben. Bezüglich der Anwendbarkeit wird untersucht, ob der Lösungsansatz nutzbar ist und dessen Gebrauchstauglichkeit (Usability) angemessen ist. Hinsichtlich des Erfolges wird geprüft, ob der Lösungsansatz die gewünschten Effekte auf die Erfolgskriterien und -faktoren erzielt. Es können hier beispielsweise Beobachtungen, Befragungen oder Simulationen zum Einsatz kommen [BC09, S. 181ff].

Das Vorgehen wird zwar aufgrund der besseren Verständlichkeit zwar linear beschrieben, ist in der Realität aber stark iterativ. So ist es möglich, im laufenden Prozess von einer Phase in eine vorherige zurückzuspringen. Von der letzten Phase kann zudem eine weitere Iteration beginnend mit einer beliebigen vorherigen Phase gestartet werden. [BC09, S. 14ff]. Die Autoren sehen sieben Arten zum Durchlaufen der DRM als möglich an, welche in Tabelle 2-1 dargestellt sind.

Es werden drei Arten zur Ausführung der jeweiligen Phasen differenziert. Die Art der Ausführung bestimmt sich durch die Art der jeweiligen Forschungsfrage und verfügbaren Zeit und Ressourcen. Bei der *literaturbasierten* Ausführung wird die Phase ausschließlich basierend auf existierender Literatur durchgeführt. *Umfassend* ergänzt die literaturbasierte Ausfühung um eigene Forschungsergebnisse, bspw. die Durchführung empirischer Untersuchungen, der Entwicklung eigener Unterstützungsansätze oder deren Evaluierung. *Initial* bedeutet hingegen, dass die Ergebnisse nur soweit aufbereitet werden, um die Konsequenzen der Forschung aufzuzeigen und diese in weiteren Phasen nutzen zu können [BC09, S. 18f].

Tabelle 2-1: Ausprägungen in der Anwendung der Design Research Methodology nach [BC09, S. 18]. Quelle: [Jop21, S. 12].

Vorgehensweisen des Forschungsdesign		Vorgehen im Forschungsdesign			
		Klärung der Forschungs-frage	Deskriptive Studie 1	Präskriptive Studie	Deskriptive Studie 2
	1	Literaturbasiert →	Umfassend	–	–
	2	Literaturbasiert →	Umfassend →	Initial	–
	3	Literaturbasiert →	Literaturbasiert →	Umfassend →	Initial
	4	Literaturbasiert →	Literaturbasiert ↑	Literaturb. / Initial / Umfassend ↑	Umfassend
	5	Literaturbasiert →	Umfassend →	Umfassend →	Initial
	6	Literaturbasiert →	Literaturbasiert ↑	Umfassend ↑	Umfassend
	7	Literaturbasiert →	Umfassend ↑	Umfassend ↑	Umfassend

2.2 Ansatz der Konsortialforschung

Ansätze wie die Design Research Methodology beschreiben zwar ein systematisches Vorgehen zur Durchführung von Design Research, berücksichtigen aber nicht explizit den Umstand, dass Forschung in einigen Bereichen wie der Wirtschaftsinformatik vorwiegend in der Praxis stattfindet. Dies greift der Ansatz der Konsortialforschung nach ÖSTERLE UND OTTO auf, welcher den Wissenstransfer zwischen Forschung und Praxis adressiert. Dabei setzt die Konsortialforschung auf Ansätzen wie der DRM auf. Der Ansatz besteht aus einer Domäne, einem Metamodell, Ergebnissen, Phasen und Aktivitäten. Er ist in Abbildung 2-2 dargestellt und soll im Folgenden näher erörtert werden [ÖO10, S. 273ff].

Die **Domäne** beschreibt das Forschungsgebiet, in dem der zu entwickelnde Ansatz genutzt werden soll. Sie umfasst das Forschungsumfeld, die betriebliche Praxis in Unternehmen sowie die Wissensbasis, welche bereits nach wissenschaftlichen Standards erarbeitetes und veröffentlichtes Wissen beinhaltet. Das **Metamodell** beschreibt die Bestandteile des zu entwickelnden Ansatzes und ihre Beziehungen untereinander [ÖO10, S. 276ff].

Der Ansatz der Konsortialforschung gliedert sich in vier Phasen. In der **Analysephase** steht die Problemidentifikation, Motivation und Zieldefinition der Lösung (Klärung der Forschungsfrage und erste deskriptive Studie der DRM) im Fokus. In der Phase wird

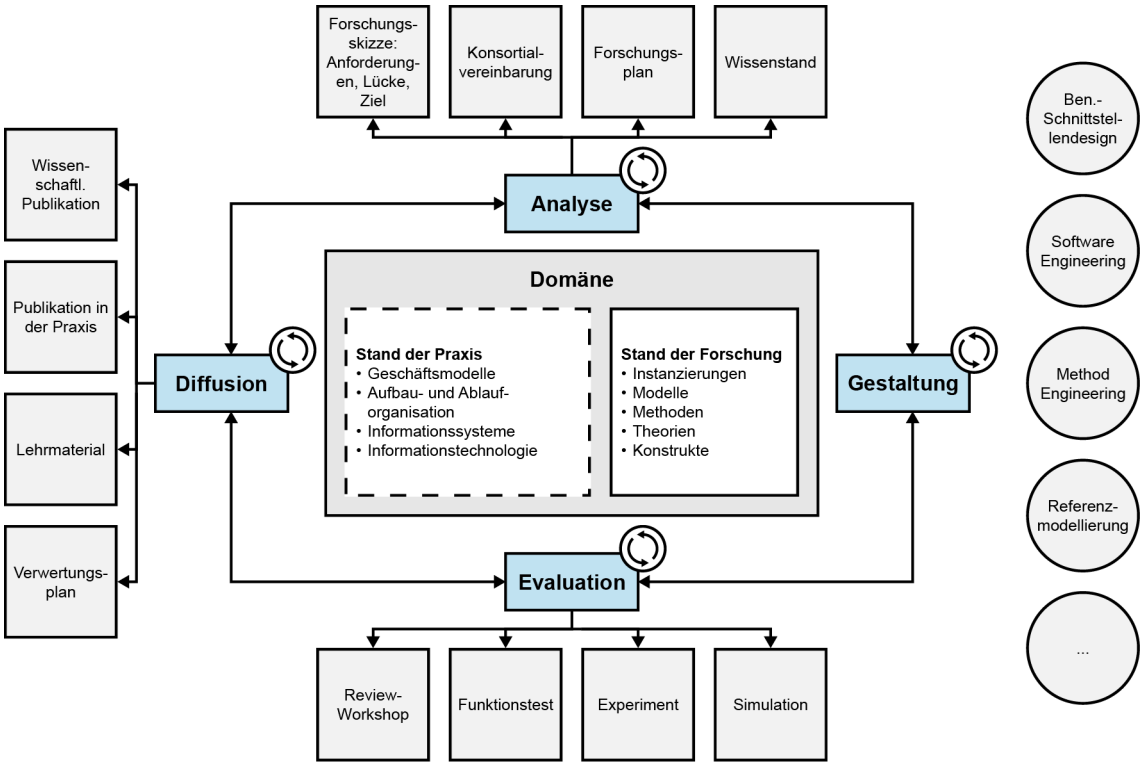


Bild 2-2: Bestandteile des Ansatzes zur Konsortialforschung nach [ÖO10, S. 278]

eine vage Forschungsidee in einem heuristischen Prozess gemeinsam detailliert und ein Konsortium aus Praxisunternehmen und Forschungseinrichtungen gebildet, die mit dem Forschungsplan einem abgestimmten Ziel folgen. In der Konsortialforschung ist besonders der Zugang zu Praxiswissen zur Ableitung von Forschungsfragen wichtig. Diese und die zur Beantwortung erforderlichen Aufwände werden durch die Praxisunternehmen geprüft. Der Forschungsplan wird so lange mit den Konsortialpartnern iteriert, bis dieser akzeptiert wird [ÖO10, S. 279].

Die nachfolgende **Gestaltungsphase** hat das Ziel, den zu entwickelnden Ansatz zu gestalten (präskriptive Studie der DRM). In der Konsortialforschung ist der Zugang zu Praxiswissen und die Relevanzprüfung zum Filtern nicht relevanter oder anwendbarer Artefakte des Ansatzes hervorzuheben. Die relevanten Artefakte werden schließlich so weit mit den Praxisunternehmen iteriert, bis sie von diesen akzeptiert werden [ÖO10, S. 279].

In der **Evaluationsphase** soll der entwickelte Ansatz gegenüber den Forschungszielen evaluiert werden (zweite deskriptive Studie der DRM). Die Evaluation der Artefakte sollte mindestens durch eine Expertenbefragung mit den beteiligten Unternehmen und einer Pilotanwendung von einem Praxisunternehmen durchgeführt werden [ÖO10, S. 279f].

Die **Diffusionsphase** umfasst die Bereitstellung der Forschungsergebnisse für die Allgemeinheit. Für die beteiligten Forschungseinrichtungen geschieht dies typischerweise über Publikationen oder im Rahmen der Lehre. Die Praxisunternehmen verbreiten die Ergebnisse innerhalb ihrer Organisation [ÖO10, S. 279f].

2.3 Einordnung und Vorgehen in dieser Arbeit

Die Analyse, Gestaltung und Evaluierung von Engineering-IT-Architekturen ist eine komplexe Aufgabenstellung. Bild 2-3 ordnet diese Tätigkeit in das Feld des Design Researchs ein. Die Durchführung dieser Tätigkeit fällt in das Feld des Designs, die Entwicklung dieser Spezifikationstechnik fällt demnach in das Feld des Design Research. Somit bietet die Design Research Methodology einen strukturierten Rahmen, welcher das Vorgehen dieser Arbeit zum Zwecke der Qualitätssicherung vorgibt.

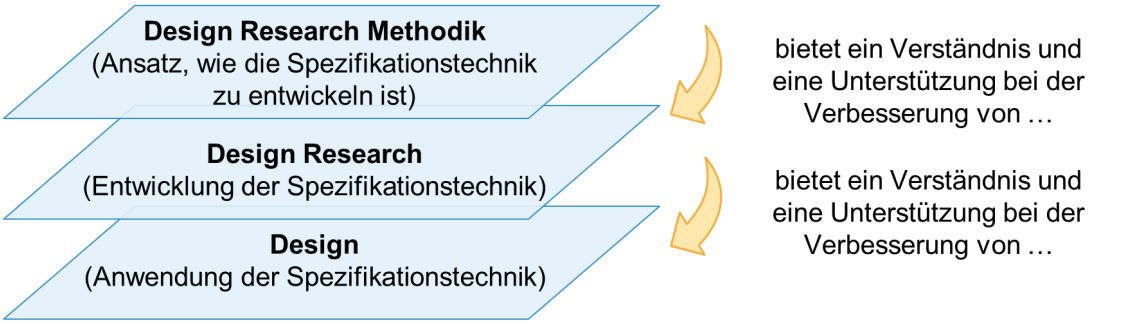


Bild 2-3: Einordnung der vorliegenden Arbeit in die Design Research Methodology nach [BC09, S. 10]. Grafik in Anlehung an [Jop21, S. 9].

Die vorliegende Arbeit orientiert sich in ihrem Vorgehen an den vier Phasen der DRM (s. Kap. 2.1). Zur Entwicklung der Arbeit wurden alle vier Phasen durchlaufen. Eine Besonderheit im Rahmen dieser Arbeit ist die Entstehung im Rahmen mehrerer Industrieprojekte mit einem Unternehmen aus der Landtechnik. Daher wird auch der Ansatz der Konsortialforschung nach OESTERLE UND OTTO herangezogen. Die Umsetzung ist in Bild 2-4 dargestellt und soll im Folgenden näher erläutert werden.

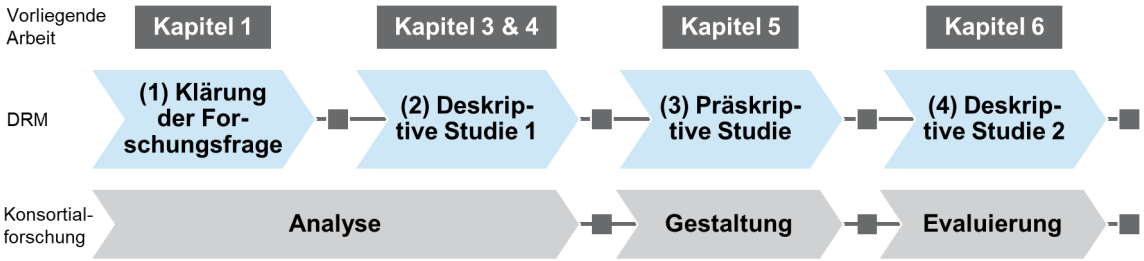


Bild 2-4: Adaptiertes Vorgehen der Design Research Methodology nach [BC09, S. 15] und der Konsortialforschung nach [ÖO10, S. 278] zur Entwicklung der Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen

Zur **Klärung der Forschungsfrage** wird primär ein literaturbasierter Ansatz gewählt. In diesem Rahmen werden Fachliteratur und aktuelle Studien herangezogen. Ergänzend dazu wird die Phase iterativ durchlaufen, da die Forschungsfrage sehr stark insbesondere in Industrie- aber auch Forschungsprojekten motiviert ist. Der Industriepartner wurde bei der Klärung der Forschungsfrage entsprechend mit einbezogen. Die sich hier aufzeichnende Forschungsfrage wird durch die Literaturarbeit bestätigt. Das Ergebnis liegt in Form der Einleitung (Kap. 1) vor.

Mit der **deskriptiven Studie 1** wird das Problemverständnis vertieft. Dazu wurde erneut eine erweiterte Literaturrecherche durchgeführt. Im Rahmen einer iterativen Anwendung an einem Beispiel aus dem Praxisunternehmen wurden zudem Problemfelder in der praktischen Anwendung identifiziert und mit den Ergebnissen der Literaturrecherche abgeglichen. Diese erlaubt es, das Themenfeld zu explorieren und die angrenzenden Themenbereiche einzuordnen (3.3 und 3.4). Daraus werden die relevanten Ansatzpunkte zur Gestaltung der Engineering-IT im Sinne von Erfolgsfaktoren hergeleitet 3.5. Basierend darauf wurden acht Anforderungen an die Arbeit abgeleitet, die im Kontext der DRM als Bewertungskriterien fungieren (Kap. 3.6). Das Ergebnis dieser Phase ist in iterativen Durchläufen mit den folgenden Phasen entstanden.

Nachdem die Bewertungskriterien festgelegt sind, erfolgt die Analyse des Stand der Technik. Hier werden existierende Ansätze basierend auf einer Literaturrecherche analysiert (Kap. 4). Durch die zuvor durchgeführte Exploration des Kontexts des Forschungsvorhabens konnten relevante Ansätze identifiziert und das Suchfeld eingegrenzt werden. Aufgrund der Vielzahl der existierenden Ansätze wurden nur solche näher im Rahmen der vorliegenden Arbeit betrachtet, welche einen Beitrag zur Erfüllung der Anforderungen leisten. Bei starker Ähnlichkeit von Ansätzen wird nur ein repräsentativer Ansatz untersucht. Fazit der Untersuchung ist, das keiner der Ansätze und auch keine triviale Kombination von Ansätzen die Anforderungen vollumfänglich erfüllen. Der Handlungsbedarf unter Berücksichtigung des Stand der Technik wurde abgeleitet, um darauf basierend eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen* entwickeln zu können.

Die Entwicklung der Spezifikationstechnik wird im Rahmen der **präskriptiven Studie** durchgeführt (Kap. 5). Dazu werden unter anderem Ansätze aus dem Stand der Technik herangezogen und entsprechend adaptiert und in die Spezifikationstechnik integriert. Die Bestandteile der Spezifikationstechnik wurden unmittelbar an praktischen Problemstellungen des Praxisunternehmens evaluiert, um nicht relevante, anwendbare oder fehlende Elemente zu identifizieren.

Im Rahmen der **deskriptiven Studie 2** wird die Arbeit angewendet und erprobt. Zur Überprüfung der Anwendbarkeit wurde das Vorgehen laufend erprobt (vgl. Kap. 6.1). Zusätzlich wurden weitere, heterogene Beispiele durch weitere Anwender aus dem Praxisunternehmen integriert, um so die Anwendbarkeit auf tatsächliche Problemstellungen in der Praxis zu überprüfen. Erkenntnisse wurden dabei durch Beobachtungen und Befragungen erlangt. Dabei stand im Fokus, dass die Spezifikationstechnik die notwendigen methodischen Anweisungen zur Durchführung der festgelegten Schritte in ausreichender Detaillierung bereitstellt und damit die erwarteten Ergebnisse erzeugt werden können. Wenn dabei Unzulänglichkeiten festgestellt wurden, wurde zurück in die vorherigen Phasen der DRM gesprungen und diese erneut durchlaufen. Dabei wurden im Rahmen der Validierung unterschiedliche Unternehmen des Landtechnikkonzerns betrachtet, um die Anwendbarkeit auch unter verschiedenen Bedingungen überprüfen zu können. Der Erfolg wurde letztlich in Hinblick auf die Anforderungserfüllung qualitativ bewertet (Kap. 7).

3 Problemanalyse

Ziel der Problemanalyse sind Anforderungen an eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*. Kapitel 3.1 definiert hierzu zentrale Begriffe, um ein einheitliches Verständnis für die nachfolgenden Kapitel aufzubauen. In Kapitel 3.2 werden zunächst die Evolution der technischen Systeme hin zu intelligenten technischen Systemen als Betrachtungsgegenstand des Engineerings und daraus resultierende Herausforderungen aufgezeigt. Kapitel 3.3 beschreibt die zur Entwicklung intelligenter technischer Systeme notwendigen Bestandteile des Engineerings und zeigt dadurch mögliche Stellhebel zur Anpassung auf.

Kern der Arbeit ist die Gestaltung und Bewertung der Informationstechnik (IT) im Engineering. Vor diesem Hintergrund geht Kapitel 3.4 auf die Aufgaben, Rollen und Gestaltungsgegenstände der Engineering-IT ein. Dazu werden typische IT-Werkzeuge näher betrachtet, Treiber für Veränderungen der Engineering-IT beschrieben und mögliche Gestaltungsoptionen für eine Engineering-IT-Architektur vorgestellt. Kapitel 3.5 grenzt das Problem ab, indem Nutzenpotentiale einer geeignet gestalteten Engineering-IT dargestellt und entsprechende Herausforderungen und Bedarfe zu deren Umsetzung diskutiert werden. Darauf basierend werden in Kapitel 3.6 die Anforderungen an die Arbeit abgeleitet.

3.1 Begriffsdefinitionen und Einordnung der Arbeit

Ziel dieser Arbeit ist eine Spezifikationstechnik, welche eine ganzheitliche Analyse und Gestaltung der Prozesse, Methoden, genutzten Artefakte und IT-Werkzeuge im Engineering erlaubt. Die wesentlichen Begriffe sollen im Folgenden für ein einheitliches Verständnis und eine bessere Lesbarkeit definiert werden.

Engineering ist nach ISO/IEC TR 19759 die Anwendung eines systematischen, disziplinierten, quantifizierbaren Ansatzes auf Strukturen, Maschinen, Produkte, Systeme oder Prozesse [ISO19759]. Nach der Definition der International Association of Engineers (IAENG) wird dabei auf wissenschaftliche Erkenntnisse, z. B. aus den Naturwissenschaften oder der Mathematik, zurückgegriffen [IAE19-ol]. Das englische Wort Engineering wird häufig synonym für die Produktentwicklung genutzt [ISO15288, S. 14f], [DAR+21, S. 25]. Im Rahmen dieser Arbeit umfasst das Engineering die vier Zyklen der Produktentwicklung nach GAUSEMEIER ET AL (vgl. Kap. 3.3.1).

Nach ABRAMOVICI ET AL. besteht ein für die Entwicklung von ITS gestaltetes Engineering aus Prozessen, Methoden, IT-Werkzeugen, Artefakten, Organisationsstrukturen und Kompetenzen [AH16, S. 23f], [TMM+22, S. 2489]. Diese Bestandteile sind in Bild 3-1 zusammenfassend dargestellt. Prozesse und Methoden beschreiben, welche Tätigkeiten wie zu durchlaufen sind. Als Resultat daraus entstehen Artefakte – Informationsobjekte, welche das zu entwickelnde System beschreiben [TMM+22, S. 2489f]. ABRAMOVICI ET AL. bezeichnen diese als Modelle. Typischerweise werden diese Artefakte mit Hilfe von

IT-Werkzeugen erstellt [Abr18, S. 5ff]. Diese können als Anwendungssoftware betrachtet werden, also Software zur Lösung fachlicher Aufgaben eines Nutzers [FDN+12, S. 92]. Die Bestandteile des Engineerings werden in Kapitel 3.3 detailliert beschrieben.

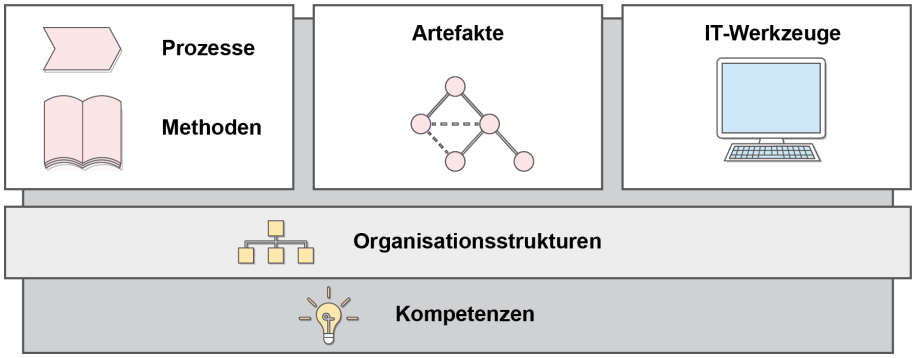


Bild 3-1: Bestandteile des Engineerings in Anlehnung an [Abr18, S. 5ff]

Zur Betrachtung komplexer Systeme wie dem Engineering eignen sich Architekturen. Die ISO/IEC/IEEE 42010 beschreibt eine **Architektur** als [ISO42010, S. 2]:

> *„fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution."*

Die Norm betrachtet eine Architektur somit als Eigenschaft eines Systems. Je nach betrachtetem System können sich demnach verschiedene Arten von Architekturen ergeben. Übertragen auf eine **Engineering-IT-Architektur** umfasst diese somit die grundsätzlichen Konzepte und Strukturen der Prozesse, Methoden, Artefakte und IT-Werkzeuge im Engineering sowie die Zusammenhänge zwischen diesen Elementen. Eine konkrete Engineering-IT-Architektur drückt somit bspw. aus, welche Prozesse und Methoden existieren, wie diese zusammenhängen und durch IT-Werkzeuge unterstützt werden können. Konzentriert man sich auf einzelne Aspekte der Engineering-IT-Architektur, können auch diese als eine (Teil-)Architektur betrachtet werden. Beispielsweise können die Strukturen der IT-Werkzeuge und ihre Beziehungen untereinander als **Applikationsarchitektur** bezeichnet werden (s. Kap. 3.4.8). Eine Architektur kann dabei einen IST oder SOLL-Zustand beschreiben.

In Abgrenzung zur Architektur, den Eigenschaften eines Systems, kann die Architekturbeschreibung eines Systems betrachtet werden. Diese wird definiert als [ISO42010, S. 2]:

> *„work product used to express an architecture."*

Während das Durchführen der Spezifikationstechnik dieser Arbeit somit die Architektur der Engineering-IT beeinflusst, ist das konkrete Arbeitsergebnis die Architekturbeschreibung in Form von Modellen. **Modelle** sind allgemein als ein Abbild eines Ausschnitts der Realität zu verstehen [Pat82, S. 306]. Nach STACHOWIAK weisen Modelle drei wesentliche Merkmale auf. Erstens sind sie immer eine *Abbildung* des Originals. Zweitens sind sie eine *Verkürzung*, da das Original abstrahiert dargestellt wird. Das Modell trägt damit zur

Vereinfachung bei, indem nur die relevanten Merkmale des Originals abgebildet werden. Drittens unterliegt ein Modell somit einem *Pragmatismus*, da es auf einen bestimmten Zweck ausgerichtet ist [Sta73, S. 131ff]. Zur Modellierung eines Systems wird nach FRIEDENTHAL ET AL. eine Modellierungssprache, eine Modellierungsmethode und ein Modellierungswerkzeug benötigt [Kai14, S. 26], [FMS12, S. 16].

Eine **Modellierungssprache** ist eine Menge von Vokabeln und grammatikalischen Regeln, um einen Sachverhalt in einem Modell abzubilden [FOS+18, S. 71]. Sie schafft eine definierte Kommunikationsgrundlage für alle beteiligten Akteure. Eine Modellierungssprache wird, wie in Bild 3-2 dargestellt, über ihre Syntax und Semantik definiert [SVE+07, S. 29f], [PM06, S. 19].

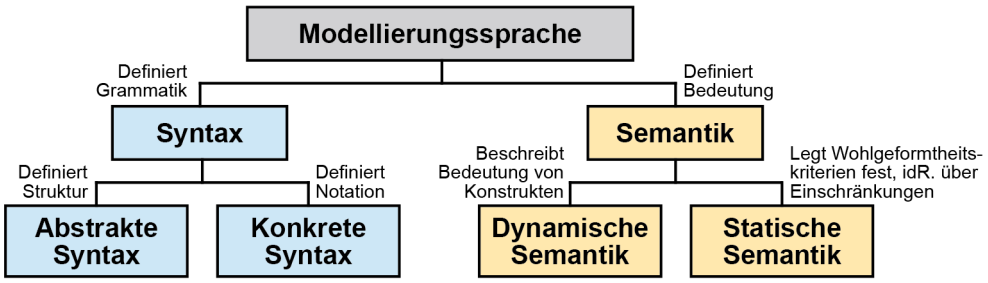


Bild 3-2: Bestandteile einer Modellierungssprache nach [SVE+07, S. 29f], [PM06, S. 19], in Anlehnung an [Ech17, S. 15]

Die abstrakte *Syntax* beschreibt die Zeichen der Sprache und ihre Beziehungen zueinander durch eine Grammatik bzw. ein Metamodell, um daraus gültige Aussagen zu bilden [SVE+07, S. 29], [KK02, S. 4], [Ech17, S. 15]. Ergänzende Bedingungen können durch formale Sprachen wie beispielsweise die Object Constraint Language (OCL) als Teil der OMG UML festgelegt werden [OMG14-ol, S. 5f]. Die konkrete Syntax legt Ausdrucksmittel und ihre Darstellungsform bzw. Notation fest [PM06, S. 19]. Sie ist wesentlich für die Akzeptanz, das Verständnis und die Effektivität der Modellierungssprache in ihrer Funktion als Kommunikationsmedium [Moo09, S. 757f].

Die *Semantik* nimmt Bezug auf den Inhalt und legt darüber die Bedeutung fest. Die statische Semantik definiert Regeln, wie einzelne Zeichen bei syntaktisch korrekter Nutzung kombiniert werden können, um eine Bedeutung zu erhalten [Tab06, S. 21], [PM06, S. 20]. Sie kann in gewissem Maße durch ein Metamodell beschrieben oder mit Hilfe der OCL oder mathematischer Ausdrücke genauer spezifiziert werden [Sod15, S. 7], [BZ14, S. 114]. Diese Regeln werden gegen die abstrakte Syntax definiert, z. B. in Form von Ontologien oder mathematischen Ausdrücken. Bspw. kann die statische Semantik Wertebereiche von Attributen oder Beziehungen zwischen Modellelementen einschränken [VAC+09, S. 267]. Die dynamische Semantik legt dann die Bedeutung der unter Beachtung der statischen Semantik gebildeten Sprachkonstrukte in Bezug auf die reale Welt fest. Dazu können spezielle formelle Beschreibungen oder auch informelle textuelle Beschreibungen genutzt werden [SVE+07, S. 30], [KK02, S. 4]. Wenn sowohl Syntax als auch Semantik (durch Logik oder ausführbar) formal beschrieben sind, liegt eine formale Modellierungssprache vor.

Eine semi-formale Sprache liegt vor, wenn zwar eine formale Syntax, aber keine formale Semantik vorliegt [Kro12, S. 109]. Formale Sprachen sind als Sonderfall anzusehen, selbst bei Programmiersprachen ist die Semantik nicht zwangsläufig formal definiert [BKL+09, S. 101].

Nach BECKER ET AL. müssen Modelle den **Grundsätzen ordnungsmäßiger Modellierung** genügen, da das Einhalten der Regeln einer Modellierungssprache nicht zwangsläufig zu einem guten Modell führen muss. Nach dem *Grundsatz der Richtigkeit* muss das Modell mit dem dem zugrundeliegenden Metamodell und dessen Notationsregeln konsistent sein (syntaktisch korrekt) und das abzubildende Objekt hinsichtlich Struktur- und Verhaltenstreue ausreichend genau sowie widerspruchsfrei wiedergeben. Der *Grundsatz der Relevanz* fordert eine in Abhängigkeit des Modellierungszweck getroffene, zweckdienliche Auswahl des modellierten Ausschnitts als Realitätsabbildung sowie ein geeignetes Abstraktionsniveau. Der *Grundsatz der Wirtschaftlichkeit* begrenzt die Modellierungstiefe, da die Modellierung betriebswirtschaftlichen Rahmenbedingungen unterworfen ist. Der *Grundsatz der Klarheit* fordert eine Anschaulichkeit des Modells für den jeweiligen Adressaten, u. a. durch Strukturiertheit, Übersichtlichkeit und Lesbarkeit. Darunter fällt auch der Umfang des Metamodells, wobei eine zu große Menge an Metamodellelementen die Klarheit beeinträchtigen kann. Der *Grundsatz der Vergleichbarkeit* legt fest, dass zwei Modelle des gleichen Sachverhalts in der gleichen Modellierungssprache miteinander vergleichbar sein sollten und in gleicher Weise den Sachverhalt widerspiegeln müssen. Zudem sollen bei unterschiedlichen Modellierungssprachen die Modelle ineinander überführbar sein [BRS95, S. 437ff], [BPV12, S. 32ff].

Eine **Modellierungsmethode** ist im Modellierungskontext eine Menge zusammenhängender Aktivitäten, Techniken und Festlegungen. Sie definiert, was spezifiziert werden muss und welche Tätigkeiten dazu in welcher Sequenz durchzuführen sind. Die Methode beschreibt, *wie* diese Schritte auszuführen sind, um daraus ein Modell und dessen Inhalte zu erstellen [FMS12, S. 21], [RFB12, S. 106]. Sie beschreibt, wie die Sprache zur Darstellung bestimmter Sachverhalte genutzt werden kann, indem sie vorgibt, wann und wie welche Diagramme oder Konstrukte einer Sprache anzuwenden sind [Alt12, S. 63], [BFP96, S. 340]. Eine definierte Methode trägt dazu bei, Missverständnisse und Kommunikationsprobleme zu reduzieren.

Ein **Modellierungswerkzeug** ist typischerweise ein IT-Werkzeug zur Erstellung, Bearbeitung, Nutzung und Verwaltung eines Modells [Alt12, S. 67ff], [Wei14, S. 403]. Die Nutzung einer spezifischen Sprache mit einem Werkzeug setzt grundsätzlich voraus, dass in diesem die entsprechende Syntax und ggf. formalisierte Semantik implementiert ist. Das Modell kann automatisiert auf korrekte Nutzung der Syntax geprüft werden und Analysen der Modellinhalte können durchgeführt werden. So können u. a. verschiedene Sichten auf das Modell erzeugt werden und Verknüpfungen zwischen den Modellelementen gepflegt werden [FMS12, S. 21], [GLL12, S. 107f].

Die drei Aspekte Sprache, Methode und Werkzeug weisen Beziehungen untereinander

auf und müssen daher aufeinander abgestimmt werden [Kai14, S. 26]. Die abgestimmte Kombination der drei Aspekte wird je nach Quelle als Methodik [RFB12, S. 106f], Instrumentarium [CJS98, S. 583] oder ohne explizite Verbindung mit einem Werkzeug auch als **Spezifikationstechnik** [Sch18, S. 11f], [Fra06, S. 8] bezeichnet.

Im Folgenden sollen die Einflussfaktoren für die Notwendigkeit einer *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen* näher beschrieben werden. Ausgangspunkt stellen dabei zunächst die im Rahmen des Engineerings zu entwickelnden Systeme dar.

3.2 Evolution Technischer Systeme

Technische Systeme unterliegen einem stetigen Wandel. Sie können in verschiedene Evolutionsstufen eingeordnet werden. Diese sind an einem Beispiel in Bild 3-3 dargestellt.

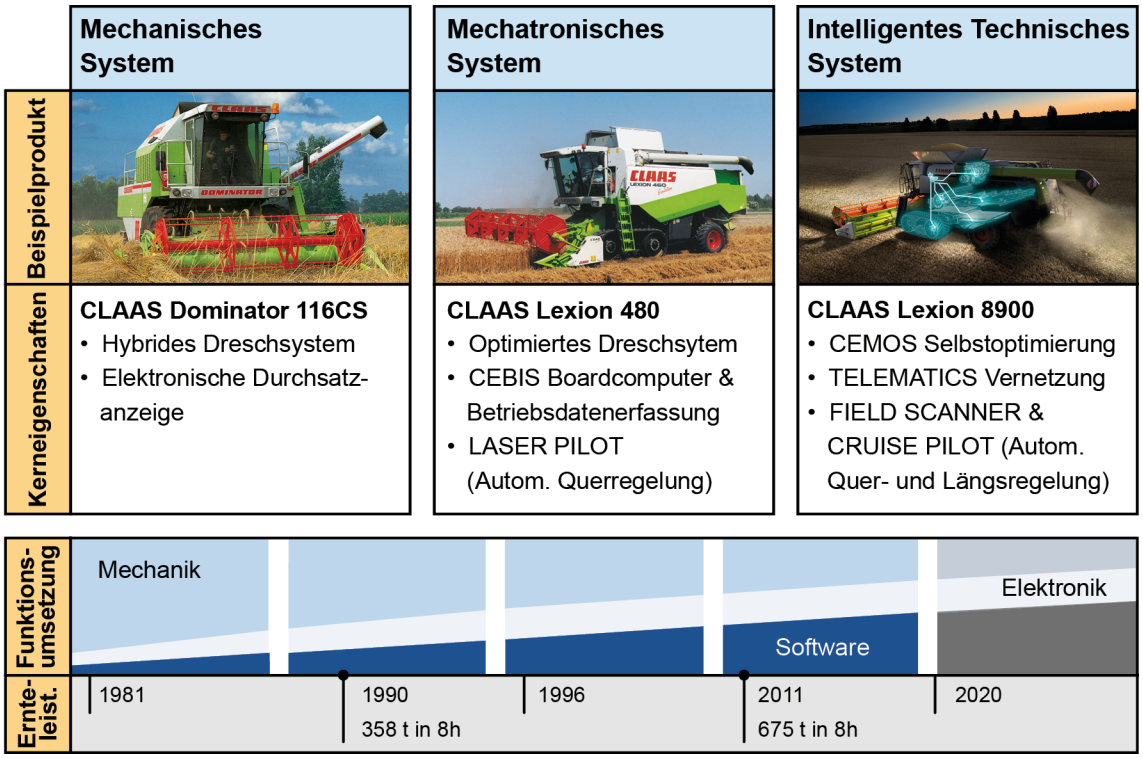


Bild 3-3: Evolution von mechanischen Systemen hin zu intelligenten technischen Systemen und ihre Auswirkungen am Beispiel der Landtechnik. Bildquelle: Claas

In vielen Systemen dominierte vor einigen Jahrzehnten noch die **Mechanik**, welche den Großteil der Funktionalität realisierte und die treibende Kernkompetenz somit auf der Disziplin des Maschinenbaus basierte. Zwar können mechanische Systeme auch elektrische Komponenten wie bspw. Elektromotoren enthalten, jedoch sind die Schnittstellen und funktionalen Zusammenhänge zwischen Mechanik und Elektrotechnik einfach. Eine integrierte Entwicklung mit hohem Maß an gegenseitigem Problemverständnis war bei mechanischen Systemen nicht notwendig [Ise08, S. 1ff], [Rod19, 4ff].

Seit den 1970er Jahren wurde die Elektronik immer stärker integriert, sodass in dieser Zeit in Japan erstmals der Begriff **Mechatronik**, als Kofferwort von *Mecha*nik und Elek*tronik* aufkam [Com94, S. 46]. Während ihrer Evolution über die letzten Jahrzehnte nahm die Digitaltechnik immer stärkeren Einfluss, sodass intelligente Steuerungen integriert wurden und die Vernetzung Einzug hielt. Dies bedingt das gegenseitige Verständnis und enge Zusammenwirken der verschiedenen involvierten Disziplinen [Rod19, S. 6]. HARASHIMA, TOMIZUKA UND FUKUDA [HTF96, S. 1f] definieren mechatronische Systeme daher als:

> *„The synergetic integration of mechanical engineering with electronic and intelligent computer control in the design and manufacturing of industrial products and processes.“*

Essentiell ist nach der VDI-Richtlinie 2206 die Emergenz, welche die synergetischen Effekte zwischen den integrierten Systembestandteilen und -technologien beschreibt (s. Kap. 3.4.7). Mechatronische Systeme bestehen in der Grundstruktur aus einem physikalischen Grundsystem, Sensoren und Aktoren sowie einer Informationsverarbeitung [VDI2206, S. 11ff]. Sie stehen miteinander in Beziehung und wirken gemeinsam in einem Regelkreis, was den hohen notwendigen Integrationsgrad der einzelnen Bestandteile widerspiegelt. Mit der Evolution der mechatronischen Systeme wuchs somit auch das Verständnis für ihre Entwicklung. Sie muss in hohem Maße interdisziplinär geführt werden, da sonst entlang der involvierten Disziplinen nicht die nötigen synergetischen Effekte aufgebracht werden können, was in Bild 3-4 dargestellt ist [Czi19, S. 5f], [Ise08, S. 3ff].

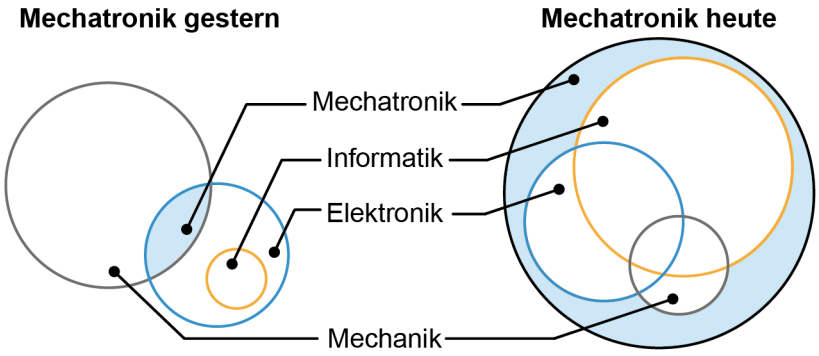


Bild 3-4: Verständniswandel und höherer Integrationsgrad bei mechatronischen Systemen nach [Eig14b, S. 43].

Mit der Evolution der mechatronischen Systeme fällt auf, dass der Anteil von Elektronik und insbesondere Software fortschreitend exponentiell wächst [Ise08, S. 6f]. Software realisiert immer mehr wesentliche Funktionen und ist dabei wichtiger Innovationstreiber [Bro10, S. 17], [Eig13, S.92]. Diese Fortschritte erlauben die Integration einer inhärenten Teilintelligenz, sodass sich ein Wandel zu **intelligenten technischen Systemen** (ITS) ergibt [DDF+14, S. 4]. ITS weisen nach DUMITRESCU folgende Kerneigenschaften auf [Dum10, S. 41]:

- Adaptivität: Interaktionsmöglichkeit mit dem Umfeld und die Möglichkeit, das Verhalten autonom an das Umfeld entsprechend dem Systemzweck anzupassen.

- Robustheit: Flexible und autonome Reaktion auch im dynamischem Umfeld, selbst auf unerwartete und vom Entwickler nicht vorhergesehene Situationen.
- Effektivität: Proaktives Verhalten, um Ziele schneller und bei Gefahrvermeidung unter Antizipation möglicher Einflüsse und Zustände zu erreichen.
- Benutzerfreundlichkeit: Berücksichtigung des spezifischen Benutzerverhaltens, indem durch Anpassung die Bedienbarkeit verbessert und das eigene Systemverhalten optimiert wird.

Damit einhergehend ist der englischsprachige Begriff der Cyber-Physical Systems (CPS) [DDF+14, S. 4], welche nach TÖRGREN UND GRONAN die Informationsverarbeitung, Vernetzung und physikalische Prozesse integrieren, um Produkte herzustellen, die autonom, intelligent, vernetzt und kollaborativ sind [TG18, S. 1]. Sie übertreffen somit die funktionalen Möglichkeiten von mechatronischen Systemen, bringen jedoch auch deutlich größere Herausforderungen bei ihrer Entwicklung mit sich. Dies ist durch die exponentiell hohe Komplexität von ITS bedingt. Die Problembewältigung durch disziplinspezifische Ansätze kommt, insbesondere bedingt durch die hohe Interdisziplinarität der Problemstellung, an ihre Grenzen und erfordert neue Entwicklungsansätze [DDF+14, S. 5]. Gleichzeitig bieten ITS in allen Bereichen unseres Lebens und Branchen der Industrie enormes Potential.

In einer Studie nach GAUSEMEIER ET AL. werden **Herausforderungen** in der Entwicklung von Produkten in der heutigen Zeit identifiziert [GDS+13, S. 16]. Insbesondere sind dies:

- Wachsende Interdisziplinarität: Für die Entwicklung von ITS sind verschiedene Disziplinen und oftmals eine Vielzahl beteiligter Personen notwendig. Jede Disziplin nutzt eigene Vorgehens- und Denkweisen sowie Dokumentationsformen. Von Beginn an müssen daher neben den Disziplinen des Maschinenbaus, der Elektrotechnik und Elektronik sowie der Softwaretechnik auch nicht-technische Disziplinen wie die Kognitionswissenschaft oder Neurobiologie einbezogen werden. Eine wesentliche Herausforderung ist daher das Sicherstellen eines einheitlichen Systemverständnisses unter allen Beteiligten, um die Integration des Systems zu ermöglichen [SD16, S. 3f], [TG18, S. 10], [GDS+13, S. 4].
- Hohe Schnittstellenvielfalt: Die zu entwicklenden Systeme können nicht isoliert betrachtet werden, da sie Schnittstellen zu ihrer Umgebung sowie zwischen den Elementen des Sytems aufweisen. ITS nutzen eine Vielzahl heterogener physischer und virtueller Schnittstellen, was durch die arbeitsteilige Entwicklung weiter begünstigt wird. Demnach müssen die Schnittstellen beherrscht und zahlreiche Wechselwirkungen beachtet werden [TG18, S. 6], [FMS12, S. 3].
- Steigende Anforderungskomplexität: Aus der hohen Komplexität und dem zunehmenden Funktionsumfang der Systeme resultieren immer mehr Anforderungen. Gegenseitige Abhängigkeiten der Anforderungen führen zu einem hohen Vernetzungsgrad und erschweren ihre Verwaltung [Ebe19, S. 247f], [FMS12, S. 3], [GDS+13, S. 17].

- Zunehmende Produktvarianz: Kunden haben zunehmend höhere Erwartungen hinsichtlich individualisierter Produkte. Die Berücksichtigung dieser Kundenwünsche führt zu einer steigenden Zahl von Produktvarianten und hoher Varianz in der Produktion (Mass Customization), die es zu handhaben gilt [MBK+19, S. 343], [Sch13, S. 92f], [HK16, S. 979].

Zusammenfassend kann festgehalten werden, dass die Komplexität der technischen Systeme steigt. ITS stellen neue Anforderungen an ihre Entwicklung. Diese muss interdisziplinär erfolgen. Zudem muss eine Vielzahl organisatorischer und technischer Schnittstellen berücksichtigt werden. Weiterhin schlägt sich die Komplexität der Systeme in einer höheren Anforderungskomplexität und Produktvarianz nieder.

3.3 Engineering technischer Systeme

Nach TÖRGREN UND GRONAN spiegelt sich die zunehmende Komplexität der technischen Systeme zwangsläufig in einer zunehmenden Komplexität ihrer Entwicklung wieder [TG18, S. 7]. Diesen Sachverhalt illustriert Bild 3-5.

Als Reaktion auf die steigende Komplexität der ITS entwickeln sich Ansätze wie das Smart Engineering [AES+12, S. 10ff] oder das Advanced Systems Engineering [DAR+21, S. 28ff]. Um ihre Ansatzpunkte genauer beleuchten zu können, sollen im Folgenden die Bestandteile des Engineerings näher erläutert werden.

3.3.1 Prozesse

Prozesse sind nach DIN EN ISO 9000 und GAUSEMEIER UND PLASS eine Menge von Aktivitäten, welche einen Input in ein definiertes Ergebnis (Output) transformieren [ISO9000, S. 33], [GP14, S. 237]. Die in einem Prozess definierten Aktivitäten beantworten somit die Frage, *Was* zu tun ist [Gad17, S. 5ff]. GAUSEMEIER ET AL. beschreiben ein Referenzmodell zur strategischen Planung und integrativen Entwicklung von Marktleistungen (4-Zyklen-Modell) [GDE+18, S. 89ff], welches den Rahmen für die dort ablaufenden Prozesse bildet. Dieses ist in Bild 3-6 dargestellt und umfasst vier iterative Zyklen. In der strategischen Produktplanung werden zunächst Potentiale identifiziert und darauf basierend eine Geschäftsidee definiert, welche in konkreten Produkt- und Dienstleistungsideen und daraus resultierende Anforderungen mündet. Anhand dieser erfolgt die Geschäftsplanung. Die Marktleistung wird integrativ in den drei Entwicklungszyklen konkretisiert, in denen das Produkt, die Dienstleistung und das Produktionssystem entwickelt werden. Dies umfasst jeweils die Konzipierung, den Entwurf und die Ausarbeitung des Produkts oder die Planung der Dienstleistung bzw. des Produktionssystems und die abschließende Integration.

ABRAMOVICI ET AL. nehmen ergänzend eine lebenszyklusorientierte Perspektive ein. So erstrecken sich die Prozesse des Engineerings über den gesamten Produktlebenszyklus (PLZ) mit den in Bild 3-7 dargestellten Lebenszyklusphasen. Die Engineering-Prozesse

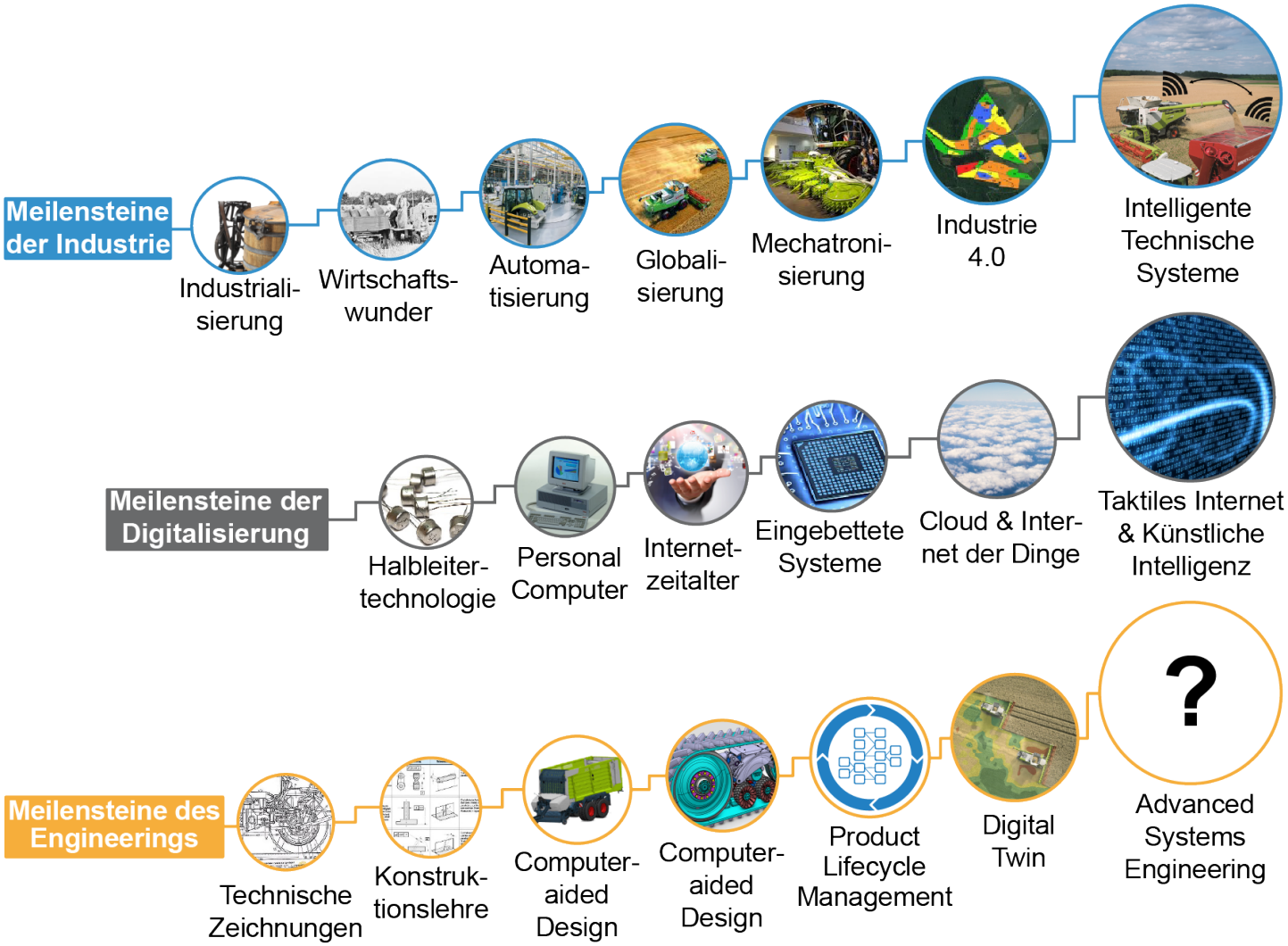


Bild 3-5: Anpassung des Engineerings an die zunehmende Komplexität technischer Systeme. Bildquellen: Claas/Fotolia

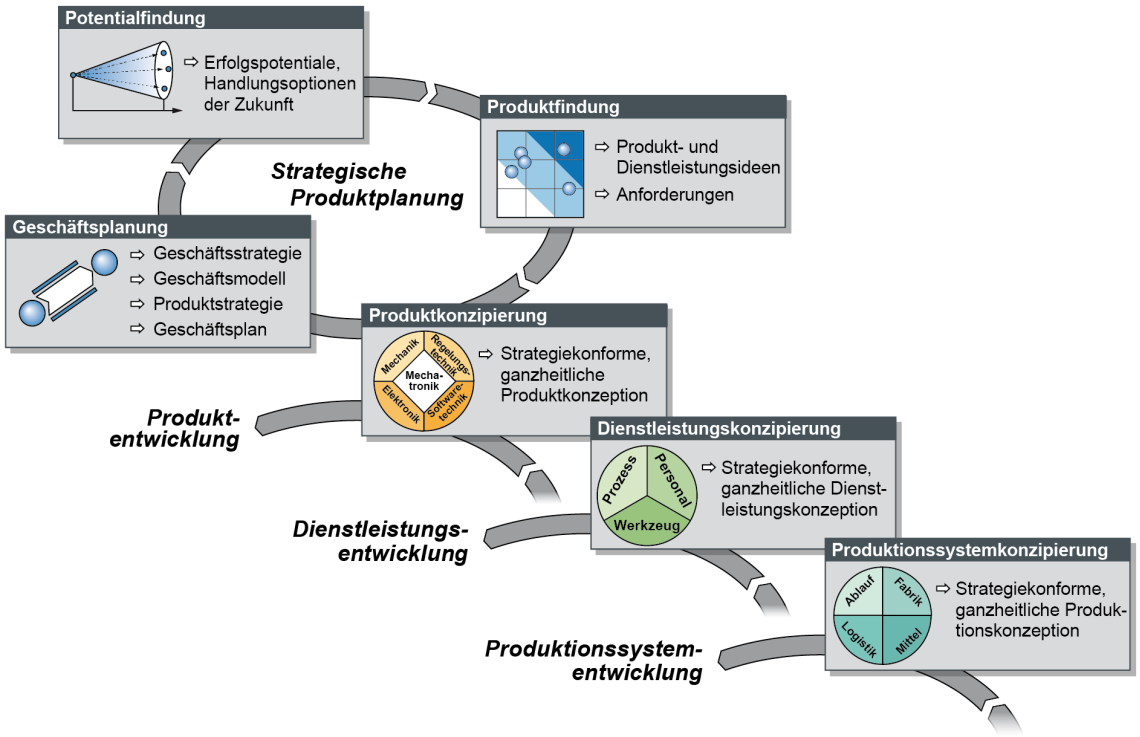


Bild 3-6: Referenzmodell zur strategischen Planung und integrativen Entwicklung von Marktleistungen [GDE+18, S. 90]

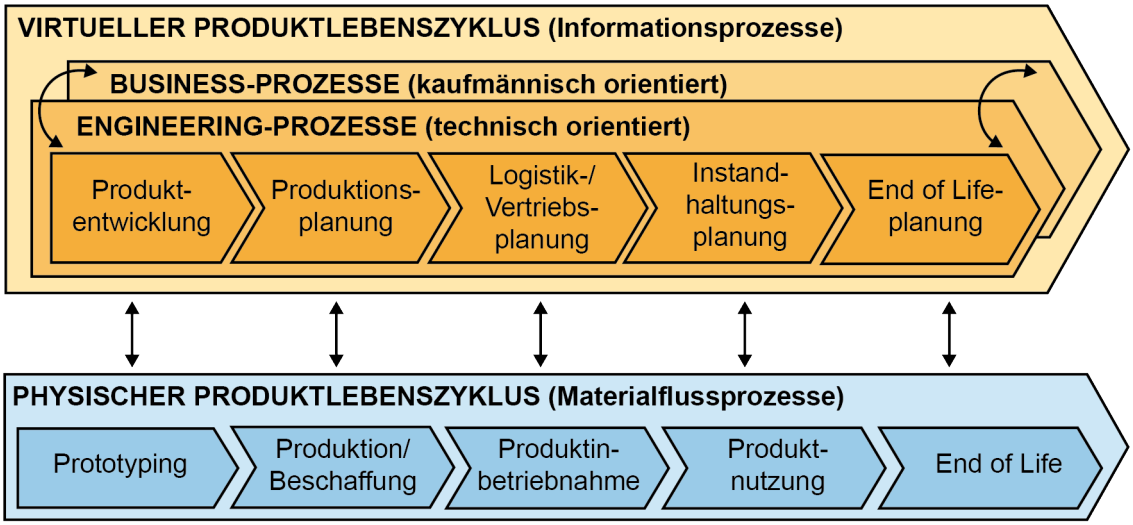


Bild 3-7: Engineering als Teil von übergreifenden Produkt- bzw. Servicelebenszyklen nach [AH16, S. 24]

sind im Wesentlichen Informationsprozesse [GP14, S. 18f] und beinhalten die technisch orientierten Aufgaben zur Planung, Definition, Konzeption, Dokumentation und Simulation von Produkten. Sie grenzen sich von den parallel ablaufenden Business-Prozessen ab, welche die kaufmännisch orientierten Aufgaben umfassen. Engineering- und Business-Prozesse bilden gemeinsam den Virtuellen PLZ. Diesem steht der physische PLZ gegenüber, der die Materialflussprozesse umfasst. Diese sind eng mit den Engineering-Prozessen verzahnt, was insbesondere im Zeitalter der Industrie 4.0 durch die zunehmenden Informationsflüsse immer relevanter wird [Abr18, S. 5f].

Engineering-Prozesse weisen im Vergleich zu den Business-Prozessen eine höhere Komplexität auf. Erstens zielen Engineering-Prozesse darauf ab, in einer einmaligen Prozesskonfiguration etwas Neues hervorzubringen, während die Business-Prozesse typischerweise einen gleichen Sachverhalt wiederholt bearbeiten. Engineering-Prozesse weisen daher Mehrdeutigkeit sowie Unsicherheit auf. Zweitens werden Engineering-Prozesse in der Regel iterativ durchlaufen, was sie synchroner, flexibler und agiler macht. Drittens sind sie stärker vernetzt. Sie werden eher in Netzwerken als in Prozessketten durchlaufen und nutzen stark vernetzte Informationsflüsse [WC18, S. 161f], [Bra13, S. 4f].

3.3.2 Methoden

Eine Methode ist nach LINDEMANN „die Beschreibung eines regelbasierten und planmäßigen Vorgehens, nach dessen Vorgabe bestimmte Tätigkeiten auszuführen sind, um ein gewisses Ziel zu erreichen“ [Lin09, S. 57]. Sie grenzt sich gegenüber einem Prozess bzw. Vorgehensmodell daher vor allem durch die formalisierte Beschreibung und den operativen Charakter ab. Eine Methode definiert daher, *wie* Prozessschritte auszuführen sind bzw. legt fest, wie die Ergebnisse zu dokumentieren sind. In unterschiedlichen Disziplinen und Kontexten werden inkonsistente Terminologien in den Methoden genutzt. Dies führt zu aufwändigen Abstimmungen zwischen den verschiedenen Disziplinen aufgrund fehlender fachdisziplinübergreifender Denkweisen und zu nicht integrierten Methoden [GDE+18, S. 384, 405], [Bra05, S. 29ff].

Methoden stellen damit häufig einen Formalismus dar, der die systematische und zielgerichtete Ausführung von einzelnen oder mehreren Prozess-Aktivitäten unterstützen soll. Methoden können dabei hierarchisch aufgebaut oder ineinander integriert werden, was jedoch komplex ist und ein tiefes Methodenverständnis erfordert. Dabei werden analog zu den Prozessen In- und Outputs genutzt, durch die verschiedene Methoden miteinander verknüpft sein können. Die Transformation des In- zu Output erfolgt durch Vorgehensschritte und ggf. unter Einsatz zusätzlicher Ressourcen wie Werkzeuge oder Vorlagen [Lin09, S. 33, 57ff].

3.3.3 IT-Werkzeuge

IT-Werkzeuge sind durch Software realisierte Anwendungssysteme (Applikationen), die den Anwender bei der Bearbeitung von Tätigkeiten unterstützen [MBK+17, S. 132], [LB15, S. 142f]. Diese Tätigkeiten können sowohl im Rahmen von Prozessen [Gad17, S. 5] als auch Methoden [Lin09, S. 75] ausgeübt werden. Anhand dieser Tätigkeiten definiert sich die notwendige Softwareunterstützung. Diese kann bis hin zu einer Automatisierung der Tätigkeiten führen.

Die Vielfalt von Prozessen und Methoden im Engineering führt in der Regel zu einer Vielzahl von Anforderungen und damit unterschiedlichen IT-Werkzeugen. Je nach Anforderungen sind die IT-Werkzeuge über Schnittstellen miteinander verbunden. Diese sind notwendig, um Daten über Werkzeuggrenzen hinweg auszutauschen und so Medien- und Informationsbrüche entlang der Prozesse und Methoden zu verhindern. Die Summe der IT-Werkzeuge und die Schnittstellen der Werkzeuge innerhalb einer Organisation bilden die IT-Infrastruktur. Sie werden typischerweise, wie in Bild 3-8 dargestellt, im Rahmen einer Applikationsarchitektur beschrieben (s. auch Kap. 3.4.8) [Abr18, S. 7], [Lei15, S. 234ff], [GDE+18, S. 382f].

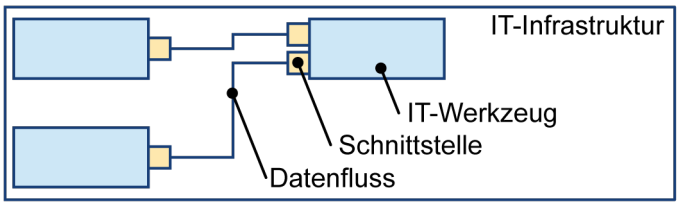


Bild 3-8: IT-Infrastruktur bestehend aus IT-Werkzeugen und Schnittstellen, dargestellt als Applikationsarchitektur

3.3.4 Artefakte

Nach *Abramovici et al.* stellen Informationsmodelle einen weiteren Bestandteil des Engineerings dar. Modelle sind im technischen Kontext nach DIN IEC 60050 mathematische oder physikalische Darstellungen eines Systems oder Prozesses mit genügend genauer Abbildung [DIN60050, S. 29]. Nach ABRAMOVICI ET AL. sind Informationsmodelle mit einer Semantik unterlegte Modelle, welche ein Produkt oder einen Ablauf im Engineering beschreiben [Abr18, S. 7]. Da sich in vielen Veröffentlichungen für digital vorliegende Arbeitsergebnisse von Prozessen und Methoden im Engineering der Begriff des Artefakts etabliert hat, soll dieser im weiteren Verlauf dieser Arbeit genutzt werden.

Im Engineering werden verschiedenste Arten von Artefakten zur Beschreibung von Systemen eingesetzt. Dies können bspw. in der Mechanik Modelle zur Beschreibung von Geometrien oder in der Elektrotechnik zur Beschreibung elektrischer Schaltungen sein. Kapitel 3.4.3 gibt einen Überblick über im Engineering verwendete Artefakt- bzw. Modelltypen und die dazu notwendigen IT-Werkzeuge. Die dedizierten Modelle stellen Partialmodelle dar, welche jeweils einen Teilaspekt beschreiben und die Struktur, das Verhalten bzw. die

Gestalt eines Systems abbilden. Beispiele für Partialmodelle können Anforderungs- oder mechanische bzw. elektrische CAD-Modelle darstellen. Werden diese zu einem kohärenten System von Partialmodellen zusammengefügt und im Laufe des Entwicklungsprozesses immer wieder verfeinert und für nachfolgende Prozessschritte genutzt, spricht man auch von einem virtuellen Prototypen oder virtuellen Produkt. Dies erlaubt es, das technische System umfänglich zu dokumentieren und Analysen sowie Simulationen durchzuführen [GP14, S. 354], [AG12b, S. 19f], [And18, S. Y18].

3.3.5 Organisationsstrukturen und Kompetenzen

Die **Organisationsstrukturen** beschreiben die Aufbauorganisation, in der die Prozesse zur Erfüllung einer Aufgabe ausgeführt werden. Die Organisationsstrukturen können dabei auf mehreren Detaillierungsebenen betrachtet werden [Abr18, S. 7]. Die Abbildung von Prozessen auf die Aufbauorganisation führt nach GAUSEMEIER UND PLASS zur Ablauforganisation. Die Aufbauorganisation hängt damit eng mit den Prozessen zusammen, da diese innerhalb der Organisationsstrukturen ausgeführt werden müssen [GP14, S. 39].

Als **Kompetenzen** werden nach ABRAMOVICI ET AL. die Fähigkeiten eines Menschens zur Lösung wissensintensiver Problemstellungen bezeichnet, womit eine persönliche Ebene betrachtet wird [Abr18, S. 11]. Wie in Bild 3-1 dargestellt, umfasst dies Kompetenzen für die Prozesse, Methoden, Artefakte, IT-Werkzeuge und Organisation im Engineering [AH16, S. 23]. Demnach fallen hierunter auch die notwendigen Fähigkeiten, um das Engineering effektiv anwenden zu können (Anwendungs- bzw. Nutzungskompetenz) [AH16, S. 24]. Von der Anwendungskompetenz ist die ebenfalls notwendige Gestaltungskompetenz zu differenzieren [Gre03, S. 90]. Diese umfasst die Fähigkeiten zur Gestaltung der Arbeitsprozesse und -umgebungen [Ger17, S. 132]. FRANK differenziert hinsichtlich der Kompetenzen und betrachtet eine Unternehmensebene, in der mehrere Einzelpersonen zusammenwirken. Eine Kompetenz wird hierbei als Kombination aus Fähigkeiten und Ressourcen[1] definiert, welche zu einem Wettbewerbsvorteil führt [Fra21, S. 20f].

3.3.6 Gestaltung des Engineerings

Die zuvor vorgestellten Bestandteile des Engineerings lassen sich in verschiedenen Ebenen strukturieren. Da das Engineering als ein Teil eines Unternehmens zu sehen ist, soll hier ein Ansatz zur **Gestaltung von zukunftsorientierten Unternehmen** zum Zweck der Strukturierung herangezogen werden. Der Ansatz von GAUSEMEIER UND PLASS wird durch das 4-Ebenen-Modell definiert [GP14, S. 37ff]. Dieses ist in Bild 3-9 dargestellt und soll im Folgenden kurz erörtert werden:

[1] Fähigkeiten ermöglichen laut FRANK die Verarbeitung von Ressourcen zur Bewältigung einer Aufgabe in Geschäftsprozessen, wohingegen Ressourcen die Qualifikationen von Personen, aber auch Werkzeuge und Instrumente umfassen.

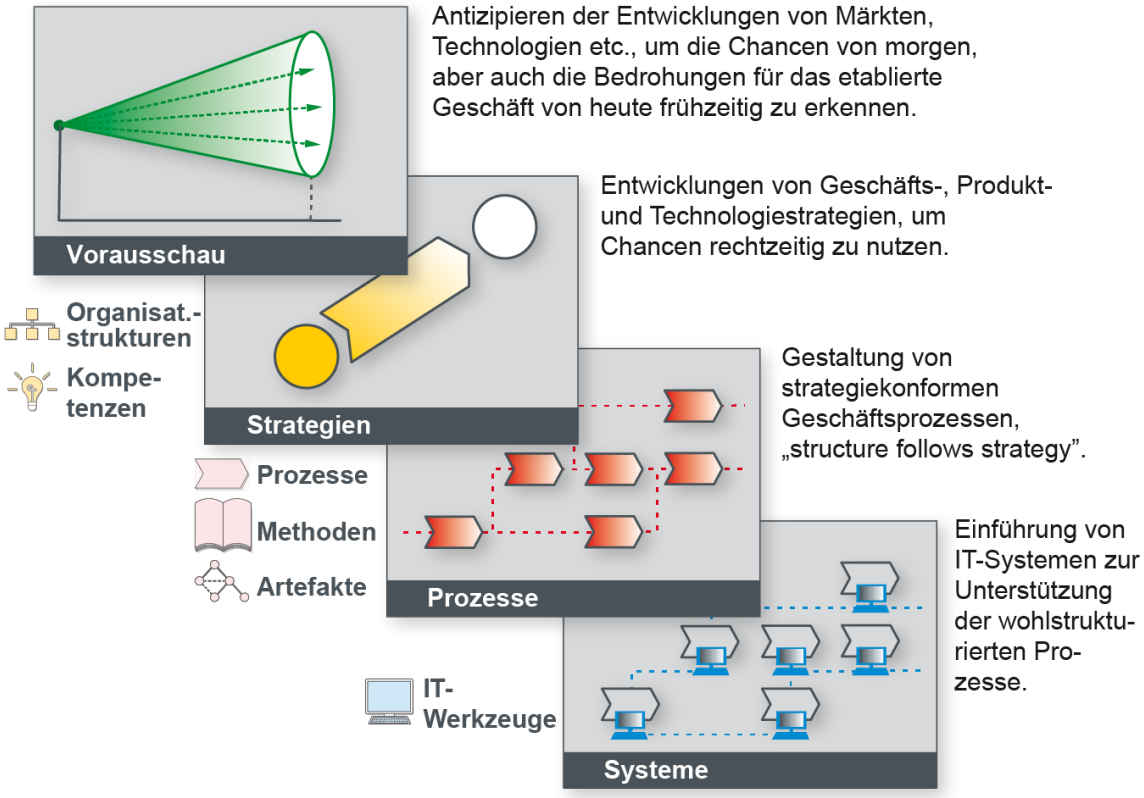


Bild 3-9: Das 4-Ebenen-Modell zur zukunftsorientierten Unternehmensgestaltung als Rahmen für die Gestaltung des Engineerings nach [GP14, S. 37ff] mit den eingeordneten Bestandteilen des Engineerings nach [Abr18, S. 5ff]

- **Vorausschau:** Der Zukunftsraum wird über Methoden wie der Szenario-Technik systematisch ausgeleuchtet, um sich abzeichnende Chancen (Erfolgs- und Nutzenpotentiale) zu identifizieren und Bedrohungen für das etablierte Geschäft zu erkennen. Sie bilden neben der Analyse der Ausgangssituation den Input für die Strategieebene.
- **Strategien:** Auf Basis der Erkenntnisse der Vorausschau wird der Kurs des Unternehmens in Form von Unternehmens- und Geschäftsstrategien festgelegt. Sie geben Leitplanken vor, in denen ausgehend vom Ausgangszustand über strategische Programme und Maßnahmen die strategischen Ziele für die einzelnen Handlungsbereiche des Unternehmens erreicht werden können. Dadurch wird eine Fokussierung auf die Vision erreicht.
- **Prozesse:** Entsprechend den Vorgaben der Strategieebene werden die Leistungserstellungsprozesse gestaltet. Sie sind die wesentliche Voraussetzung für die effiziente Erstellung einer Leistung. Sie stellen somit das Kernelement für die Definition der Ablauforganisation dar, welche sich aus der Projektion der Prozesse auf die Aufbauorganisation ergibt.
- **Systeme:** Basierend auf der Prozessebene werden die IT-Systeme definiert. Diese umfassen neben IT-Werkzeugen als Anwendungssoftware (s. Kap. 3.3.3) auch die zu deren Betrieb notwendige Basisinfrastruktur, bestehend aus Hardware (z. B. Server-

und Netzwerksysteme) und Systemsoftware (z. B. Betriebs- und Datenbanksysteme). Ziel ist die Unterstützung der Aufgaben aus der Prozessebene, weshalb eine enge Verzahnung zwischen den Bedarfen auf Prozess- und den Möglichkeiten auf Systemebene notwendig ist.

Der **Gestaltungsraum des Engineerings** ergibt sich aus der Einordnung der Bestandteile des Engineerings (s. Bild 3-1) in das 4-Ebenen-Modell (s. Bild 3-9). Die Ebenen der Vorausschau und Strategie münden letztlich in verschiedenen Ebenen von Strategien. Diese gehen von der Untenehmensstrategie über die Geschäftsstrategien hin zu den Substrategien, z. B. für das Engineering. Die Strategien geben die Leitplanken und strategische Ziele vor [GP14, S. 113ff]. Die Strategieebene beeinflusst die Organisationsstrukturen (Aufbauorganisation) und die notwendigen Kompetenzen (Kernkompetenzen) (s. Kap. 3.3.5) [GP14, S. 141, 202]. Die wesentliche Gestaltung des Engineerings findet somit innerhalb der gegebenen Leitplanken auf der Prozess- und Systemebene statt. Die Prozessebene beschreibt die durch sie definierten Leistungserstellungsprozesse des Engineerings (s. Kap 3.3.1) und die deren Durchführung unterstützenden Methoden (s. Kap 3.3.2). Darüber hinaus umfasst sie die in den Prozessen und Methoden entstehenden Artefakte (s. Kap. 3.3.4). Die Gestaltung der Systemebene fokussiert die IT-Werkzeuge (s. Kap. 3.3.3) und die Basisinfrastruktur.

Verschiedene Studien haben den **Handlungsbedarf in Bezug auf das Engineering** im Kontext von ITS bzw. CPS untersucht. TÖRNGREN UND GRONAN haben hierzu mehrere Studien untersucht. Sie führen als bestehende Optionen zur Komplexitätsbeherrschung prozess-basierte Ansätze, modell-basierte Ansätze, IT-Werkzeuge, methodische Ansätze über das Design und die Architektur sowie die Anpassung von Kompetenzen und der Organisationen auf. Die Analyse kommt zu dem Schluss, dass ein Bedarf neuer Methoden, IT-Werkzeuge und Wissen für zukünftige ITS besteht [TG18, S.9ff].

Dies spiegelt sich auch in einer davon unabhängigen Studie von ABRAMOVICI ET AL. [AH16, S. 24ff] wider. Bild 3-10 zeigt eine Zusammenfassung der Teilergebnisse einer quantitativen Befragung. Über die Hälfte der Befragten verfolgen Strategien zur Anpassung der Organisationsstrukturen. Weiterhin wird ein Bedarf zur Erweiterung der Kompetenzen gesehen, insbesondere hinsichtlich der methodischen Fähigkeiten als auch bzgl. der Prozess-, Organisations- und IT-Werkzeug-Kompetenzen. Hinsichtlich der Prozessebene wird davon ausgegangen, dass bereits solide Grundlagen existieren. Die Mehrheit der Befragten hält die Methoden des Systems Engineering (80 %) und des Product Service Systems Engineering (63 %) für grundsätzlich geeignet. Nur 20 % bzw. 37 % sehen die Ansätze als nicht geeignet an. Lediglich indirekt wurde die Bewertung der Prozesse durchgeführt. So wurde über die Methoden z.T. auch die Eignung der Prozesse mitbewertet. Innerhalb der Kategorie der Methoden wurden von 37 % der Befragten generische Prozessmodelle gefordert. Zudem existieren im Rahmen des Systems Engineering bereits generische Prozesse [ISO15288, S. 19ff]. Letztlich werden zwar dediziert für die Prozesse weitere Handlungsbedarfe genannt, diese stellen jedoch überwiegend Optimierungen dar, die sich über existierende Optimierungsansätze, wie der Geschäftsprozessoptimierung oder

Business Process Reengineering [Gad17, S. 32ff], umsetzen lassen. Die Artefakte wurden nicht explizit bewertet, es wird aber eine hohe Datenvielfalt herausgestellt. Die heutigen IT-Werkzeuge sehen lediglich 34 % als geeignet an, 61 % hingegen als eher oder gar nicht geeignet.

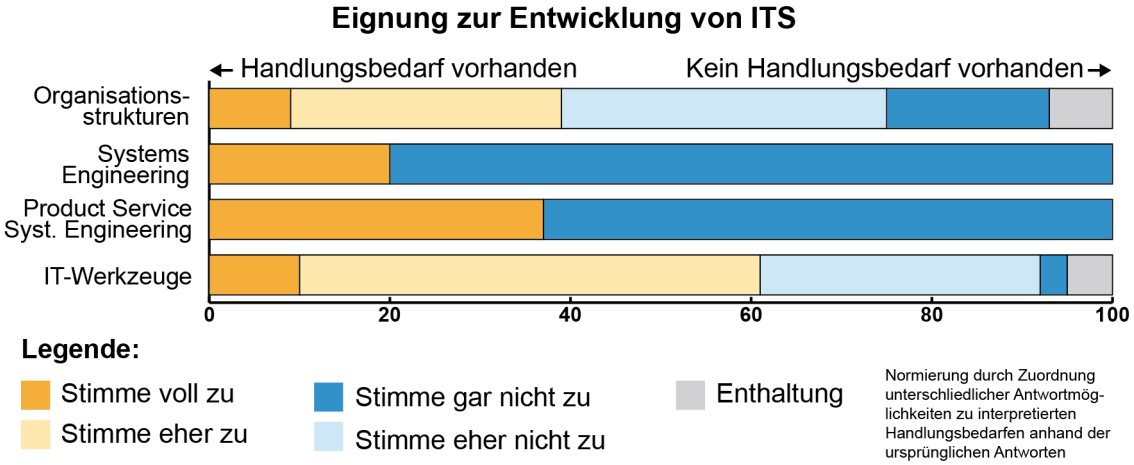


Bild 3-10: Einschätzung des Handlungsbedarfs im Engineering hinsichtlich der Entwicklung intelligenter technischer Systeme (ITS) nach [AH16, S. 24ff]

Zusammenfassend kann festgehalten werden, dass als Reaktion auf die sich verändernden technischen Systeme neue Entwicklungsansätze wie das ASE entstehen. Diese nehmen Einfluss auf die verschiedenen Bestandteile des Engineerings, welche in das 4-Ebenen-Modell nach GAUSEMEIER et al. eingeordnet wurden. *Organisationsstrukturen* (Aufbauorganisation zur Ausführung der Prozesse) und die *Kompetenzen* (Fähigkeiten zur Lösung von Problemstellungen) werden der strategischen Ebene zugeordnet und daher nachfolgend nicht weiter betrachtet. Der Strategieebene folgend kann die Prozessebene, bestehend aus *Prozessen* (sachlogische Abfolgen von Aktivitäten), *Methoden* (beschreiben regelbasiert die Durchführung von Prozessaktivitäten) und *Artefakten* (zweckdienliche Darstellungen eines Systems, z. B. zur Dokumentation), gestaltet werden. Die Systemebene ist auf die Unterstützung der Prozessebene ausgerichtet und umfasst *IT-Werkzeuge* (IT-Applikationen, welche die Durchführung von Prozessen und Methoden oder Erstellung und Repräsentation von Artefakten erlauben). Im Bereich der IT-Werkzeuge wird ein hoher Handlungsbedarf zur Anpassung an die Entwicklung von ITS gesehen. Dazu ist jedoch eine Analyse der Prozessebene und daraufhin optimierte Gestaltung der IT-Werkzeuge im Engineering notwendig.

3.4 Engineering-IT

Die Engineering-IT soll als Teil des Engineerings im Folgenden über ihre Aufgaben definiert (Kap. 3.4.1) und typische involvierte Rollen vorgestellt werden (Kap. 3.4.2). Es werden gängige IT-Werkzeuge im Engineering (Kap. 3.4.3) sowie deren Anpassungsmöglichkeiten an eine Engineering-Organisation (Kap. 3.4.4) und Integrationsmöglichkeiten der Werkzeuge untereinander (Kap. 3.4.5) aufgezeigt. Darauf basierend werden Treiber,

die zur Veränderung der Engineering-IT führen, dargelegt (Kap. 3.4.6). Zur systemischen Betrachtung der Engineering-IT wird in die Systemtheorie und das (Enterprise) Systems Engineering eingeführt. Abschließend werden Gestaltungsansätze für die ganzheitliche Planung der Engineering-IT-Infrastruktur anhand von Enterprise Architekturmodellen diskutiert (Kap. 3.4.8).

3.4.1 Aufgaben der Engineering-IT

Die Engineering-IT stellt Dienstleistungen für die im Rahmen des Engineering benötigte IT-Infrastruktur bereit und umfasst die zu ihrer Planung und ihrem Betrieb verbundenen Personen und Tätigkeiten. Dies beinhaltet sowohl strategische als auch operative Aufgaben, welche zyklisch durchlaufen werden und zur Abgrenzung nachfolgend anhand eines generischen Modells erläutert werden sollen.

Die Information Technology Infrastructure Library (ITIL) beschreibt ein verbreitetes Referenzmodell für das Management von IT-Dienstleistungen innerhalb einer Organisation. Es umfasst die Entwicklung und Bereitstellung von IT-Services[2] [Krc15, S. 608], [FOS+18, S. 36]. IT-Services müssen dabei dem Kriterium der Nützlichkeit (Utility) und der Zuverlässigkeit (Warranty) genügen. ITIL umfasst fünf in Bild 3-11 dargestellte Phasen. Die Phase der *Service Strategy* umfasst die strategische Planung und Definition der IT-Organisation und nimmt Einfluss auf alle weiteren Phasen. Sie definiert die Anforderungen an das Design, die Übergabe und den Betrieb. Die Phase *Service Design* beinhaltet die Planung und Konzeption der IT-Services. In der Phase *Service Transition* wird der IT-Service bis zum laufenden Betrieb eingeführt und bestehende Services angepasst. Der Service-Betrieb zur ständigen und nachhaltigen Erbringung des IT-Services geschieht in der Phase *Service Operation*. In den vorherigen Phasen gesammelten Erfahrungen werden in der kontinuierlich mitlaufenden Phase des *Continual Service Improvements* aufgenommen und zur Verbesserung der IT-Services genutzt [Lei15, S. 208f], [HH11, S. 49].

3.4.2 Rollen in der Engineering-IT

In die Planung und den Betrieb der Engineering-IT ist eine Vielzahl von Stakeholdern involviert. Entlang der Phasen werden im Folgenden Aufgaben identifiziert und generische Rollen zugeordnet. In der Phase der Service Strategy Definition müssen die Anforderungen an den IT-Service definiert werden [Gri10, S. 86f]. Hier können die späteren **Anwender** der IT-Services wichtige Angaben zu ihren Bedarfen tätigen [MBK+17, S. 132]. Sie sind bspw. System-, Domäne- oder Validierungsingenieure oder Projektmanager und nutzen die IT-Services im Rahmen ihrer Tätigkeiten in Prozessen und Methoden [Lu19, S. 6], [Lin09, S. 63]. Sie besitzen die dafür notwendige Anwendungs- bzw. Nutzungskompetenz. Der Anwender hat ein hohes Interesse, seine Engineering-Tätigkeiten möglichst

[2] Aus Gründen der Lösungsneutralität wird von der Bereitstellung von IT-Services gesprochen, was im Kontext der Engineering bspw. die betriebsfähige Bereitstellung einer CAD-Anwendung sein könnte.

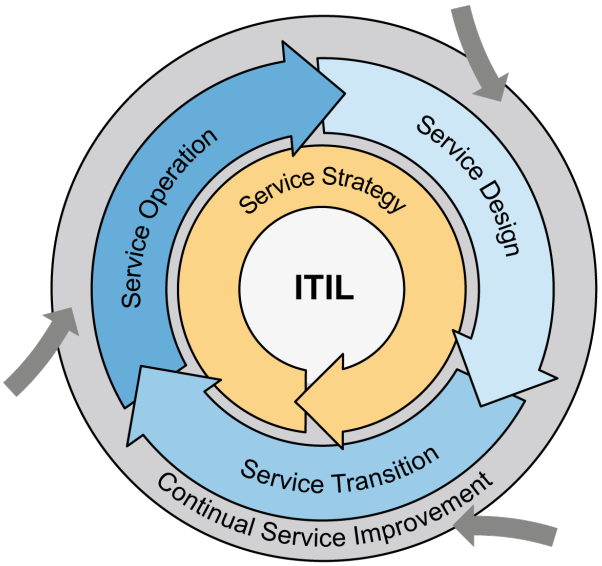


Bild 3-11: Die fünf Phasen nach ITIL zur Planung und Bereitstellung von IT-Services in Anlehnung an [Lon12, S. i]

effizient durchführen zu können. Er kann daher besonders gut Bedarfe und Feedback zur Gebrauchstauglichkeit geben.

Der **Prozess- und Methodenspezialist** kann ebenfalls bei der Anforderungsdefinition maßgeblich unterstützen. Er definiert und integriert Prozesse bzw. Methoden in die bestehende Engineering-Umgebung auf der Prozessebene [Lin09, S. 63]. Er weist eine Gestaltungskompetenz für die Prozesse und Methoden auf. Dadurch kann der Prozess- und Methodenspezialist Aussagen zu den notwendigen Funktionen treffen, da ihm die Wirkzusammenhänge der Prozesse und Methoden und damit auch die notwendige Werkzeugunterstützung bekannt sind. Sein primäres Interesse ist daher typischerweise, die Prozesse und Methoden effektiv mit einem IT-Service umsetzen zu können. Da die Artefakte den In- und Output für Prozesse und Methoden darstellen, kann der Prozess- und Methodenspezialist aus fachlicher Sicht die Bestandteile und Beziehungen der Artefakte beschreiben.

Der **IT-Spezialist** zeigt sich für die Planung und operative Realisierung des IT-Services auf Grundlage der definierten Anforderungen verantwortlich. Er kann daher je nach Rollenaufteilung innerhalb der Engineering-IT in allen fünf Phasen tätig werden. Ein Schwerpunkt liegt neben der Anforderungsdefinition mit den Stakeholdern in der Service Design Phase. Hier müssen die Bedarfe aus der Prozessebene analysiert und mittels Applikationen und Basisinfrastruktur umgesetzt werden. Dies umfasst insbesondere die Konzipierung und Implementierung der Anwendungssoftwaresysteme. IT-Spezialisten sind dabei neben der Inbetriebnahme auch für die laufende Wartung der Systeme zuständig [GP14, S. 344ff], [Lu19, S. 6].

Im Rahmen des Service Designs und der Service Transition kommt häufig Standardsoftware von **Software-Vendoren** zum Einsatz. Die Software-Vendoren sind Spezialisten

für die von ihnen angebotene Software, der Kontext des Kunden ist häufig nur teilweise bekannt. Der Software-Vendor hat das Ziel, eine Menge von Anforderungen der Service Strategy bestmöglich zu erfüllen, teilweise sind hierzu auch individuelle Anpassungen notwendig. Daher ist ein genaues Verständnis der Anforderungen relevant [Krc15, S. 205ff], [MBK+17, S. 134, 153ff], [TW15, S. 244ff].

Jede der genannten Rollen kann aus unterschiedlichen Domänen (Ingenieurwissenschaften, Informationstechniker), Kontexten (Entwicklungsabteilung, Fachabteilung für die Engineering-Prozesse und Methoden, IT-Services) und Unternehmen stammen. Dies hemmt potentiell die Kommunikation und das gemeinsame Verständnis. Gleichzeitig ist jedoch zur erfolgreichen Abarbeitung der Phasen eine enge Zusammenarbeit unabdingbar.

3.4.3 IT-Werkzeuge im Engineering

Die im Engineering genutzten IT-Werkzeuge sind sehr vielfältig. Grundsätzlich kann zwischen zwei Typen von IT-Werkzeugen unterschieden werden. Autorensysteme unterstützen die Durchführung einzelner Engineering-Aufgaben und generieren oder manipulieren dabei den Inhalt von Artefakten, wie bspw. CAD- oder CAE-Werkzeuge. Verwaltungssysteme hingegen verwalten ganze Abläufe im Engineering und die darin entstehenden Artefakte und ihre Meta-Daten, wie PDM- oder TDM-Systeme [Ger16, S. 215]. Sie unterscheiden sich primär hinsichtlich ihrer Art und Funktionalität. Diese kann sich je nach Disziplin bzw. Domäne unterscheiden. Zum Teil decken einige IT-Werkzeuge auch mehrere Arten und Funktionen ab. Daher werden im Folgenden die Werkzeuge hinsichtlich ihrer Funktionalität differenziert in Anlehnung an EIGNER ET AL. [ERZ14] vorgestellt und in Tabelle 3-1 zusammengefasst.

Mit Beginn der Entwicklung werden domänenübergreifende oder -spezifische Anforderungen dokumentiert und entlang ihres Lebenszyklus verwaltet. Dies wird auch als **Requirements Engineering** (RE) bezeichnet. Klassischerweise werden Anforderungen textuell beschrieben, sie können jedoch auch in Form von Systemmodellen visuell modelliert werden [RS14, S. 185ff]. Eine Markt-, Produkt- oder Komponentenanforderung beschreibt ein Bedürfnis oder Nutzen und kann den Problem- und Lösungsraum umfassen [Ebe19, S. 21ff]. Neben der Dokumentation von Anforderungen und ihren Attributen[3] muss ein geeignetes IT-Werkzeug auch das Requirements Management umfassen. Dies beinhaltet die Bildung von hierarchischen Strukturen, die Verwaltung von Abhängigkeiten, den Informationsaustausch, das Workflowmanagement zur kontrollierten Statusänderung von Anforderungen oder Anforderungskatalogen und die Verwaltung ihrer Lebenszyklen [RS14, S. 367ff]. Typische IT-Werkzeuge sind eigenständige Anwendungen wie bspw. IBM DOORS oder Siemens Polarion Requirements oder in PLM (z. B. Dassault Systèmes ENOVIA) oder ALM Lösungen (z. B. Microsoft Team Foundation Server) integrierte Applikationen.

[3] Attribute sind zu einem Objekt gehörige Eigenschaften, die durch Attributwerte beschrieben werden können [OB09, S. 281f].

Tabelle 3-1: Arten von IT-Werkzeugen im Engineering gegliedert nach Funktionalität

Funktionsbereich	Kernfunktionen	Beispielwerkzeuge
Requirements Engineering (RE)	Dokumentation und Verwaltung (idR. textueller) Anforderungen	IBM Doors; Siemens Teamcenter (PDM) bzw. Polarion (CASE)
System-spezifikation (MBSE)	Domäneübergreifende, ganzheitliche Modellierung von Strukturen oder Verhalten (z.B. in SysML)	SparxSystems Enterprise Architect, NoMagic Cameo Systems Modeler (CSM)
System-simulation (MBSE)	Domäneübergreifende, ganzheitliche Simulation von Systemen und Konzepten (z.B. mit Modelica, Matlab)	Mathworks Matlab/Simulink (propr. Sprache); DS Dymola (Modelica); CSM (SysML)
Mechanik-konstruktion (M-CAD)	Modellierung von Geometrien und Erzeugung von Konstruktionsdaten	DS CATIA, PTC Creo, Siemens SolidEgde
E/E-Konstruktion (E-CAD)	Entwurf von Schaltungen aus elektr. Betriebsmitteln (EK), Leiterplatten- (EDA) und Chipdesign (ICD)	EPLAN Electric P8, Mentor Graphics Capital (EK); Cadence Virtuoso (EDA, ICD)
Softwareentwicklung (CASE)	Analyse, Entwurf (U-CASE); Coding, Test, Debugging (L-CASE); Verwaltung (I-CASE)	NoMagic MagicDraw (U-CASE); MS Visual Studio (L-CASE); IBM Rational (I-CASE)
Komponenten-simulation (CAE)	Analyse-, Simulations-, Festigkeitsrechnungsverfahren für Mechanik, E/E	DS Simulia (domänespez.); dSPACE (HIL/SIL)
Digitale Fabrik (DiFa)	Durchgängiges Datenmanagement und Simulation in der Produktion	DS DELMIA, Siemens Tecnomatrix
Produktdaten-management (PDM)	Speicherung und Verwaltung der produktbezogenen Daten und Strukturen	DS ENOVIA, PTC Windchill, Siemens Teamcenter

Eng zusammenhängend mit dem RE ist die **Modellierung von Systemspezifikationen**. Sie ist in der Regel domänenübergreifend, kann aber auch domänespezifisch, z. B. in Form von Elektronikarchitekturen, erfolgen. Diese Modelle beschreiben im Kern Anforderungen oder Lösungsentscheidungen. Sie betrachten idR. das gesamtheitliche System oder dessen Elemente und werden daher auch dem Model-based Systems Engineering (MBSE) zugeordnet [Zaf14, S. 77ff], [WRF+15, S. 181f]. Häufig wird zur Modellierung die von der Object Management Group (OMG) spezifizierte Sprache Systems Modeling Language (SysML) genutzt, wobei jedoch auch bspw. proprietäre Sprachen zum Einsatz kommen können (s. Kap. 4.1.2). Die SysML erlaubt die Erstellung visueller Struktur-, Verhaltens-, oder Anforderungsdiagramme [OMG19-ol, S. 211f]. Die hierzu genutzten IT-Werkzeuge erlauben die Erstellung und Bearbeitung von Modellen. Darüber hinaus können je nach IT-Werkzeug der Lebenszyklus der Modelle und die Abhängigkeiten von Modellelementen mittels Beziehungen verwaltet sowie die erstellten Modelle ausgewertet werden. Beispielhafte IT-Werkzeuge sind der SparxSystems Enterprise Architect oder der NoMagic Cameo Systems Modeler.

Die **Modellierung zur Systemsimulation** weist starke Abhängigkeiten zur Modellierung der Systemspezifikation auf. Sie wird zur domänenübergreifenden Simulation von Syste-

men genutzt, häufig um noch in der Entwurfsphase Konzepte abzusichern. Dazu werden Modelle genutzt, die je nach Simulationszweck auch aus der Systemspezifikation stammen können. Wesentlicher Unterschied ist die Ausführbarkeit dieser Modelle, um das Verhalten des Systems analysieren zu können [WRF+15, S. 180f], [Sta22, S. 568f]. Neben der SysML kommen als Modellierungssprache, insbesondere bei multiphysikalischen Simulationen, Modelica oder andere proprietäre Sprachen zum Einsatz [GZ14, S. 107]. Beispielhafte IT-Werkzeuge sind Matlab/Simulink von MathWorks (propr. Sprache), Dassault Systèmes Dymola (Modelica) oder NoMagic Cameo Systems Modeler (SysML).

Wo die zuvor vorgestellten Themenfelder nahezu alle Domänen und Disziplinen involvieren und das System ganzheitlich betrachtet wird, sollen im Folgenden domänen- bzw. disziplinspezifische Themenfelder betrachtet werden. Weit verbreitet und bekannt ist die **Konstruktion der Mechanik** mit Hilfe von Computer-Aided Design (CAD). Dabei werden primär Geometrien modelliert, welche sowohl zwei- (2D) als auch dreidimensional (3D) ausgeführt sein können. Neben der geometrischen Modellierung umfasst die Konstruktion die Erzeugung von Konstruktionsdaten, u. a. das Ableiten von Zeichnungen, die Erstellung grafischer Werkstückmodelle und sowie von Stücklisten [Heh11, S. 120], [And18, S. Y18ff]. Beispielhafte Werkzeuge sind Dassault Systèmes CATIA, PTC Creo oder Siemens SolidEdge.

In der **Konstruktion der Elektrik und Elektronik** können verschiedene hierarchische Betrachtungsebenen eingenommen werden. Die *Elektrokonstruktion* beschreibt die logische Verschaltung und örtliche Platzierung von aus Bauteilen zusammengesetzten Komponenten (z. B. Sensoren, Aktoren, Steuerungen oder elektrischen Leitern) zur Realisierung von elektrischen Installationen, z. B. in Maschinen, Anlagen oder Gebäuden. Dabei werden u. a. Schaltpläne, Verdrahtungspläne oder Stücklisten erzeugt [Zic13, S. 11, 74ff]. Darüber hinaus können auch Bauräume innerhalb von CAD-Modellen für die Leitungsführung definiert werden. Beispielhafte IT-Werkzeuge stellen z. B. EPLAN Electric P8 oder Teile der Capital Suite von Mentor Graphics dar. Im *Leiterplattenentwurf* werden diskrete Bauelemente wie Widerstände oder Chips auf einer Leiterplatte platziert und über die Pins der Bauteile mittels in den Leiterplatten befindlicher Leiterbahnen verbunden [FL12, S. 211f]. Dabei werden Schaltpläne und physische Layouts in Verbindung mit den entsprechenden Bauteildaten und Platzierungsdaten genutzt, welche oft auch in der Produktion Verwendung finden [Rei07, S. 160f]. Die hier Anwendung findenden IT-Werkzeuge werden oft als Electronic Design Automation (EDA) Werkeuge bezeichnet, bspw. Teile der Suiten Cadence Virtuoso oder Zuken CR-8000. Beim *Chipentwurf* werden integrierte Schaltkreise gestaltet, welche monolithische Funktionseinheiten darstellen, die physisch nicht trennbar sind. Sie werden in Layouts gezeichnet, welche für den Fertigungsprozess benötigte Fotomasken darstellen. Dabei müssen durch den Fertigungsprozess vorgegebene Design Rules beachtet werden [Rei07, S. 574f]. Beispielhafte Werkzeuge sind z. B. Teil der Cadence Virtuoso Suite.

Die **Softwareentwicklung** wird durch Computer-Aided Software Engineering (CASE) unterstützt. CASE umfasst alle IT-gestützten Hilfsmittel zur Verbesserung der Software-

Qualität, der Entwicklungs-Produktivität und des Entwicklungs-Managements. CASE Werkzeuge können dabei einzelne Tätigkeiten unterstützen und sich zu einer CASE-Plattform integrieren. Diese stellt Verwaltungs-Funktionen zur Datenhaltung, Versions- und Benutzerverwaltung bereit [Rau07, S. 163f]. Upper-CASE Werkzeuge unterstützen die frühen Phasen, also die Analyse, das Software RE und den Entwurf, was häufig in Form von Diagrammen nach der OMG Unified Modeling Language (UML) modelliert wird. Lower-CASE Werkzeuge unterstützen Aufgaben in den späteren Lebenszyklusphasen, insbesondere Coding, Test und Debugging [Kur08, S. 341ff]. Werden beide Bereiche abgedeckt und sind Funktionen zum Management des gesamten Softwarelebenszyklus (Konfigurations- und Änderungsmanagement) und von Softwareprojekten vorhanden, spricht man von Integrated-CASE (I-CASE) [Rau07, S. 164] oder auch vom Ansatz des Application Lifecycle Management (ALM). ALM ist als das Gegenstück von PLM zur Softwareentwicklung zu sehen und umfasst demnach Prozesse, Arbeitsweisen und Werkzeuge [KV08, S. 56f]. Beispielhafte Werkzeuge sind NoMagic MagicDraw (Upper-CASE), Microsoft Visual Studio (Lower-CASE) oder die IBM Rational Suite (I-CASE).

Die **Modellierung zur Komponentensimulation** im Rahmen des Computer-Aided Engineering (CAE) kann domänespezifisch als auch -übergreifend durchgeführt werden. Sie ermöglichen eine frühzeitige Produktoptimierung noch vor dem Bau physischer Prototypen. Es existieren zahlreiche genutzte Analyse-, Simulations- und Festigkeitsrechnungsverfahren, welche abhängig von der Zielstellung sind. Beispiele für domänespezifische CAE-Verfahren sind in der Mechanik die Finite-Elemente-Methoden (FEM), Mehrkörpersimulationen (MKS), Computational Fluid Dynamics (CFD), Noise-Vibration-Harshness (NVH) und das Digital Mock Up (DMU) [ES09, S. 51]. In der Elektro- bzw. Softwaretechnik werden bspw. Electrical Rule Check (ERC) oder Software in the Loop (SiL) genutzt. Domänenübergreifende Verfahren kommen bspw. beim Test von Embedded Systems, bestehend aus Elektronik und Software, in Form von Hardware in the Loop (HiL) zum Einsatz, welche nicht nur Software, sondern auch kompatible Testhardware voraussetzen [Rou14, S. 175ff]. Beim Einsatz von CAE-Werzeugen bilden sich Werkzeugketten, da die CAE-Werkzeuge idR. auf andere Entwicklungsartefakte, bspw. CAD-Daten in der Mechanik, zugreifen, anhand dieser Berechnungen durchführen und als Resultat zu verwaltende Berechnungsdaten ausgeben. Beispielhafte Software Produkte sind bei Dassault Systèmes in der SIMULIA Familie zu finden, dSPACE ist ein bekannter Software- und Hardwarelieferant für SiL und HiL Lösungen.

Um entwickelte Produkte effizient fertigen zu können, werden **Modelle und Simulation in der Produktion** genutzt. Die Digitale Fabrik (DiFa) ist hier Oberbegriff für die digitalen Modelle, Methoden und Werkzeuge, welche zu einem durchgängigen Datenmanagement integriert werden. Dabei werden häufig 3D-Visualisierung und Simulationen genutzt, mit dem Ziel, die reale Fabrik in Verbindung mit den zu produzierenden Produkten ganzheitlich zu planen, zu evaluieren und laufend zu verbessern. Hierbei kommen Computer Aided Manufacturing (CAM) zur rechnerbasierten Steuerung von Fertigungs- und Montagemaschinen, Manufacturing Process Planning Management (MPM) zur Erstellung von

Produktionsplänen, Werkzeuge zur Simulation und Optimierung von Fabriklayouts und -logistik sowie Rapid Prototyping (RPT) zum Einsatz [ES09, S. 53ff], [BGW18, S. 9ff, 26ff]. Dazu werden häufig Produktdaten aus der Entwicklung mit spezifischen Produktionsdaten verknüpft. Beispielhafte IT-Werkzeuge sind bei Dassault Systèmes in der DELMIA und bei Siemens in der Tecnomatrix Familie zu finden.

Das **Produktdatenmanagement** (PDM)[4] erlaubt es, die Vielzahl der das Produkt beschreibenden Entwicklungsartefakte zu verwalten. PDM beinhaltet die datenbankbasierte Speicherung der Artefakte (Dokumentenmanagement und Archivierung) und häufig die Stamm-, Struktur- und Stücklistenverwaltung sowie das Konfigurations-, Änderungs-, Freigabe- und Workflowmanagement [ES09, S. 31ff]. Verwaltungssysteme, die sehr nahe an den Autorenwerkzeugen arbeiten und häufig mit deren Funktionen integriert sind, werden als Team Data Management (TDM) Systeme bezeichnet. Dabei ist es möglich, dass ein Autorenwerkzeug mit einem TDM System kombiniert ist. Weiterhin können TDM Systeme die entsprechenden Schnittstellen aufweisen, um mit PDM Systemen gekoppelt zu werden und damit eine übergreifende Verwaltung und Steuerung zu erlauben [ES09, S. 43f]. Das PDM ist damit eine der wichtigsten Anwendungen im Engineering.

Werden IT-Werkzeuge, Prozesse und die darin anfallenden Informationen über den gesamten PLZ im Rahmen eines strategischen Ansatzes integriert, spricht man von **Product Lifecycle Management** (PLM) [Ger16, S. 215f]. Es grenzt sich vom reinen PDM durch den höheren Integrationsgrad und die in Bild 3-12 dargestellte breitere Abdeckung des PLZ ab [ES09, S. 37]. PLM ist somit ein integrierendes Konzept, das es erlaubt, die im Engineering anfallenden Artefakte mit Hilfe definierter und dokumentierter Prozesse über den gesamten PLZ hinweg zu organisieren und zu verwalten. Dazu werden funktionsübergreifende PDM-Systeme genutzt, um die PLM-Prozesse entlang des gesamten PLZ zu unterstützen. PLM bezeichnet daher keine in sich geschlossene IT-Lösung, sondern integriert typischerweise einzelne Systeme zu einer Gesamtlösung [Sta15, S. 3], [ADE+11, S. 10f]. Gleichwohl werden integrierende IT-Lösungen auf Basis eines PDM-Systems auch stellenweise als PLM-Lösung oder PLM-System bezeichnet [ES09, S. 39], [ADE+11, S. 11].

Insbesondere zur Unterstützung der Materialflussprozesse werden Systeme zur **Produktionsplanung und -steuerung** (PPS) eingesetzt, zu denen das Engineering Schnittstellen aufweist. PPS umfasst die gesamte technische Auftragsabwicklung entlang der gesamten Liefer- und Wertschöpfungskette, beginnend mit der Angebotsbearbeitung bis zur Lieferung des produzierten Gutes an den Kunden [SSR12, S. 4f]. Dabei werden nicht nur die Fertigung, sondern auch vor- und nachgelagerte Bereiche wie die Beschaffung, Finanz- und Personalwesen und Logistik mit einbezogen. Ein wesentliches IT-System ist hier das Enterprise Ressource Planning-System (ERP), welches der Planung und Steuerung betrieblicher Ressourcen dient. Dabei weisen ERP-Systeme Schnittmengen und Schnittstellen mit den PDM-Systemen auf, insbesondere bei der Verwaltung von Artikelstammdaten. Weiterhin kommen Manufacturing Execution Systeme (MES), welche echtzeitfähig zwi-

[4] Häufig wird auch der englischsprachige Begriff Product Data Management verwendet.

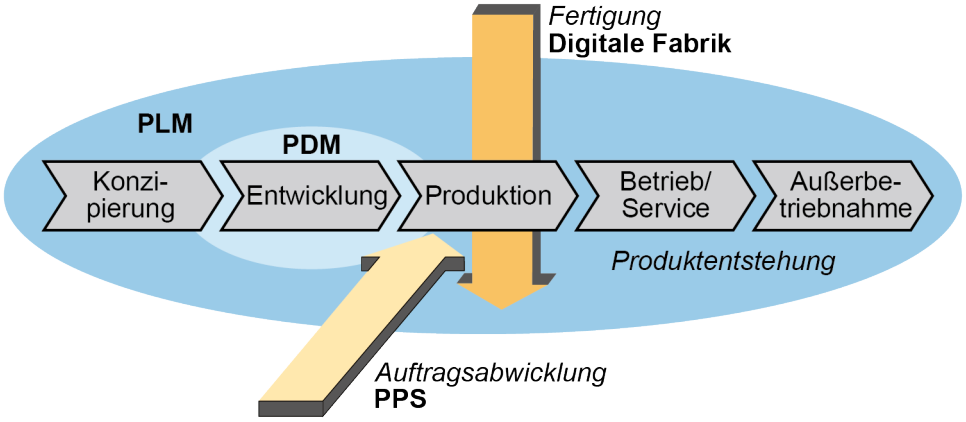


Bild 3-12: Ansätze und zugehörige IT-Werkzeugtypen zur umfassenden Unterstützung der Hauptgeschäftsprozesse nach [GP14, S. 32], [Eig14a, S. 270] und [WRF+15, S. 29]

schen Maschinensteuerung und ERP-System nah am Fertigungsprozess agieren. Customer Relationship Management Systeme (CRM) unterstützen Marketing-, Vertriebs- und Serviceentscheidungen. Supply Chain Management (SCM) Werkzeuge unterstützen bei der Planung und Steuerung des Wertschöpfungsnetzwerk, meist über PPS oder ERP Systeme [MRE14, S. 302ff, 310ff], [BGW18, S. 195ff]. Beispielsweise integriert SAP S/4HANA ERP, SCM und CRM, während Dassault Systèmes APRISO ein MES System darstellt.

Bild 3-12 fasst Ansätze zur umfassenden Unterstützung der drei Hauptgeschäftsprozesse Produktentstehung, Fertigung und Auftragsabwicklung eines produzierenden Industrieunternehmens zusammen. Die Hauptgeschäftsprozesse weisen untereinander Informationsbeziehungen auf, welche durch Schnittstellen zwischen den betroffenen IT-Werkzeugen unterstützt werden [GP14, S. 18]. In der Darstellung werden nur Konzepte berücksichtigt, die einen der Hauptgeschäftsprozesse möglichst vollständig umfassen.

3.4.4 Anpassungsmöglichkeiten von IT-Werkzeugen im Engineering

IT-Werkzeuge weisen unterschiedliche Profile hinsichtlich ihrer Eignung für die unternehmensspezifischen Anforderungen aus der Prozessebene auf [Krc15, S. 203ff]. Um implementierbare IT-Werkzeuge bereitzustellen, existieren zwei grundsätzliche Typen von IT-Werkzeugen. Es lässt sich zwischen Standardsoftware (im Englischen auch als Commercial of the Shelf (COTS) bezeichnet) und Individualsoftware unterscheiden [SK14, S. 224ff].

Standardsoftware wird für eine Gruppe von Kunden mit ähnlichen Problemstellungen als implementierbare Software am Markt angeboten. Voraussetzung für ihren Einsatz ist die grundsätzliche Erfüllung der Anforderungen aus der Prozessebene durch die Software, wobei in den meisten Bereichen Standardsoftware genutzt wird [MBK+17, S. 132]. Standardsoftware bietet sich daher zur Unterstützung von Prozessen an, die für alle Unternehmen ähnlich sind [LLS16, S. 904]. Sie bietet den Vorteil, dass keine Entwicklungs-

kosten anfallen und die Software durch den Hersteller laufend gewartet wird. Weiterhin werden Entwicklungsrisiken weitgehend vermieden und es werden weniger Ressourcen zur Implementierung benötigt. Nachteilig ist jedoch, dass die Software häufig nicht alle unternehmensspezifischen Anforderungen abdecken kann. Zudem können Schnittstellenprobleme auftreten, sofern das IT-Werkzeug nicht alle Anforderungen selbst erfüllen kann [SK14, S. 134f], [Lei15, S. 239]. Der Zustand einer nicht angepassten Standardsoftware wird auch als “Out-of-the-Box“ (OOTB) bezeichnet [Eig14a, S. 292].

Sollte die Standardsoftware in der angebotenen Ausprägung nicht den Anforderungen genügen, kann durch **Anpassungen** eine bessere Abdeckung der Anforderungen erreicht werden [MBK+17, S. 132]. Dabei existieren zwei zu unterscheidende Verfahren, die hier als *Parametrisierung* (Vom Softwarehersteller vorhergesehene Anpassungsmöglichkeiten) und *Customizing* (Anpassung über Veränderung des Programmcode oder des Datenbankschemas) definiert werden [ES09, S. 347f], [MBK+17, S. 132]. Häufig werden an Engineering-IT-Werkzeugen Anpassungen zur Funktionserweiterung, der Abbildung unternehmensspezifischer Prozesse und Workflows sowie der Anpassung von Schnittstellen zur Integration verschiedener IT-Werkzeuge vorgenommen [Eig14a, S. 291]. Nachteilig ist beim Customizing die Releaseabhängigkeit, sodass bei Versionswechseln Änderungen ggf. neu übertragen werden müssen und so hohe Wartungskosten verursachen [MBK+17, S. 136]. Darüber hinaus sind bereits die initialen Kosten enorm hoch und übersteigen typischerweise die Lizenzkosten deutlich und können sogar die genannten Kostenvorteile wieder aufheben [Lei15, S. 240]. Wie in Bild 3-13 ersichtlich entfallen nach Angaben der UNITY AG 71 % der Kosten einer PLM-Einführung auf die Anpassung. Wiederum 75 % der Anpassungskosten sind für die Erfüllung funktionaler Anforderungen notwendig, lediglich 13 % werden für die Umsetzung von Integrationsanforderungen genutzt. Bei der Einführung von PDM-Systemen sind nahezu immer Anpassungen notwendig, bei reinen TDM-Systemen ist teilweise die Nutzung von OOTB-Software möglich [Eig14a, S. 291f].

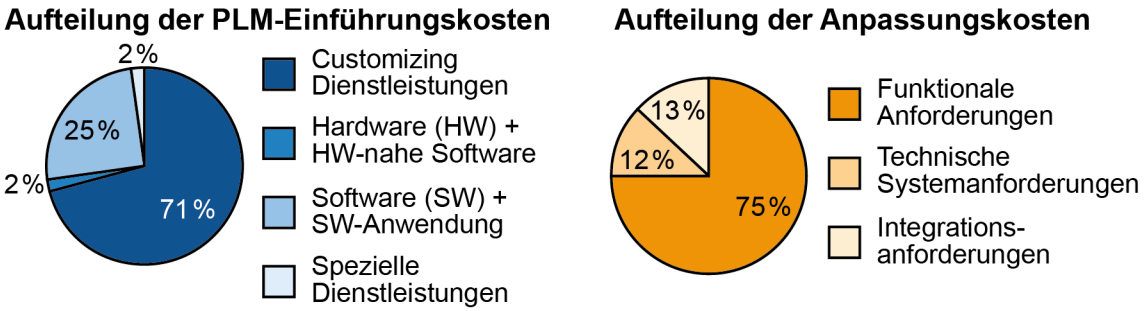


Bild 3-13: Aufteilung der gesamten PLM-Einführungskosten und der darin enthaltenen Anpassungskosten in Anlehnung an [Eig14a, S. 292]

Neben der Nutzung von Standardsoftware existiert die Möglichkeit zur Erstellung einer **Individualsoftware**. Diese ist eine speziell mit Hinblick auf die Anforderungen der Organisation durch die eigene IT-Abteilung oder beauftragte Dritte entwickelte Software. Sie hat daher den Vorteil, dass sie perfekt auf die Anforderungen abgestimmt werden kann. Nachteilig sind jedoch die hohen Entwicklungskosten und -risiken [MBK+17, S. 18, 135],

[SK14, S. 135]. Aufgrund des hohen Ressourcenbedarfs kommen Individualentwicklungen häufig nur für größere Unternehmen in Frage [Lei15, S. 238]. So existieren im Bereich der PDM-Systeme bei einigen Automobilherstellern individuelle PDM-Systemlösungen wie SMARAGD bei Daimler oder KVS bei Volkswagen. Diese Systeme sind jedoch in der Regel extrem angepasste IT-Werkzeuge, die im Kern auf Standardsoftware basieren, z. B. im Fall von SMARAGD auf dem PDM-System Metaphase, das heute in Produkten von Siemens PLM Software aufgegangen ist [ES09, S. 255], [BH98, S. 9].

3.4.5 Integration der IT-Werkzeuge im Engineering

Die IT-Werkzeuge im Engineering entwickeln sich stetig weiter. Dabei lässt sich eine Integration der Funktionen bzw. IT-Werkzeuge beobachteten. Diese Evolution der Integration soll im Folgenden näher erläutert werden und ist in Bild 3-14 zusammengefasst. Durch die Integration soll eine Interoperabilität zwischen IT-Werkzeugen erreicht werden, mit dem Ziel, Daten möglichst nahtlos zwischen ihnen auszutauschen und eine Kollaboration der IT-Werkzeuge zu ermöglichen [DB11, S. 2].

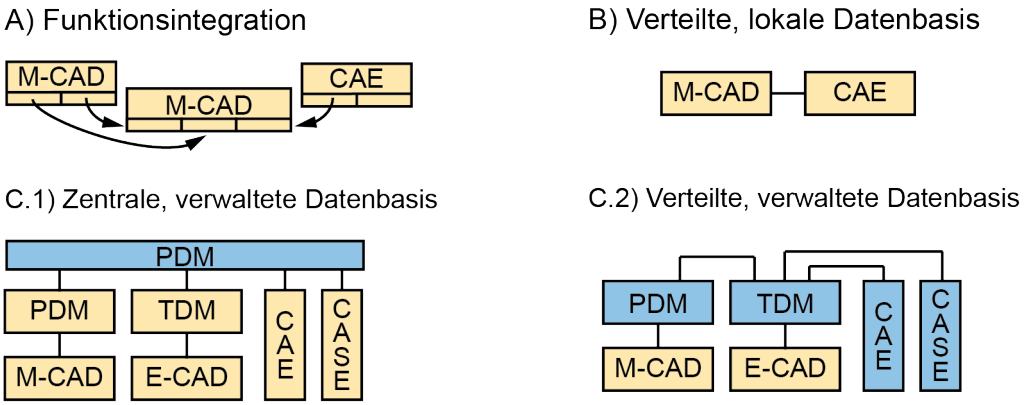


Bild 3-14: Verschiedene Arten der Integration von Funktionen aus Engineering-IT-Werkzeugen

In der einfachsten Form lässt sich eine **Vermischung der Funktionsbereiche** aus Kap. 3.4.3 beobachten. So wurden bspw. in M-CAD Werkzeuge auch Funktionen integriert, welche nicht primär der geometrischen Modellierung dienen. Dies kann bspw. Funktionen zur numerischen Simulation (CAE) oder zur automatisierten Fertigung (CAM) umfassen [ES09, S. 49], [Heh11, S. 118f].

Zunehmend erkennbar ist auch die **Bildung von Werkzeugketten**. Wie bereits angedeutet sind im Engineering eine Vielzahl von Prozessabläufen mittels ihrer In- und Outputs über den gesamten PLZ miteinander verzahnt. Dies setzt auch eine entsprechende Integration und Durchgängigkeit seitens der Daten und Informationen zwischen den Werkzeugen voraus, um die Aufgaben effektiv bearbeiten zu können [VDI3695-4, S. 3f]. Zum reibungslosen Austausch von Artefakten sind Schnittstellen zwischen den Werkzeugen erforderlich, weshalb es Bestrebungen zur Standardisierung von Austauschformaten und Schnittstellen gibt. Zur Bildung einer Werkzeugkette werden zwei oder mehr IT-Werkzeuge über Aus-

tauschformate oder Schnittstellen miteinander gekoppelt, mit dem Ziel, insbesondere die Datenqualität und -verfügbarkeit zu erhöhen. Dies geschieht häufig nur in Teilbereichen des Entwicklungsprozesses [DR13, S. 231]. Eine beispielhafte Werkzeugkette erlaubt, dass mechanische Komponentensimulationen in einem CAE-Werkzeug auf Basis von 3D-CAD-Modellen aus einem CAD-Werkzeug durchgeführt werden können. Dies verringert den Aufwand zur Modellerstellung für die Komponentensimulation deutlich [RNW16, S. 905]. Aufgrund der Vielzahl an Werkzeugen mit einem eigenen Datenbestand wird in diesem Fall von *verteilten, lokalen Datenbeständen* gesprochen. Eine gesteuerte Verwaltung der Artefakte erfolgt in diesem Fall nicht.

Die Integration von Werkzeugketten schlägt sich in den Konzepten **PLM und ALM** nieder. Dabei sollen die Informationen über den gesamten Lebenszyklus eines physischen Produktes (PLM) oder einer softwarebasierten Applikation (ALM) konsistent verwaltet werden. In diesem Zusammenhang werden Werkzeugketten immer großflächiger integriert. Eine Strategie ist die Integration von Funktionen in einem umfassenden IT-Werkzeug (integrativer Ansatz). In der Praxis nutzen die Produkte eine *zentrale, verwaltete Datenbasis* und werden von einem Hersteller mit einer monolithischen Architektur angeboten. Da der Hersteller die Kontrolle über die Schnittstellen besitzt, können die Werkzeuge daher effektiver integriert und die Artefakte konsistent verwaltet werden [FNF+14, S. 214ff], [ES09, S. 257]. Beispiele hierfür finden sich z. B. bei Dassault Systèmes mit der 3DEXPERIENCE Plattform zur Unterstützung des PLM-Ansatzes oder Siemens POLARION zur Unterstützung des ALM-Ansatzes. Erfüllt eine monolithische Lösung jedoch nicht sämtliche an sie gestellten Anforderungen vollumfänglich, muss typischerweise doch auf externe Schnittstellen und weitere IT-Werkzeuge zurückgegriffen werden [SK14, S. 135].

Als gegensätzliche Strategie gilt der föderierte Ansatz, bei dem ein Netzwerk von IT-Werkzeugen über Schnittstellen entsteht [AZL19, S. 9]. Ziel ist eine *verteilte, verwaltete Datenbasis*. Diese wird von den präferierten IT-Werkzeugen gebildet und verwaltet, man ist daher nicht auf einen bestimmten Anbieter beschränkt. Die Abstimmung und Wartung der Schnittstellen ist jedoch in der Praxis eine große Herausforderung. Insbesondere die konsistente Verwaltung der Artefakte, beispielsweise mittels Konfigurationsmanagement, gestaltet sich hier schwierig [FNF+14, S. 218], [ES09, S. 257].

Schnittstellen sind zur Integration der IT-Werkzeuge essentiell. Nach SINDERMANN ergeben sich wie in Bild 3-15 dargestellt zwei wesentliche Dimensionen zur Charakterisierung von Schnittstellen. Zum einen betrifft dies die *Informationsübertragung*. Bei einer direkten Informationsübertragung werden die Daten mit Hilfe geeigneter Schnittstellen ohne die Nutzung von Zwischenformaten zwischen IT-Werkzeugen übertragen. Bei der indirekten Informationsübertragung werden die zu übertragenden Daten im Quellsystem in einer Datei mit einer spezifischen Syntax und Semantik gespeichert und im Zielsystem interpretiert oder abgelegt [Sin14, S. 328].

Weiterhin können Schnittstellen hinsichtlich der *Offenheit* der verwendeten Syntax und Semantik als Datenformatspezifikation differenziert werden. Ein neutrales Datenformat

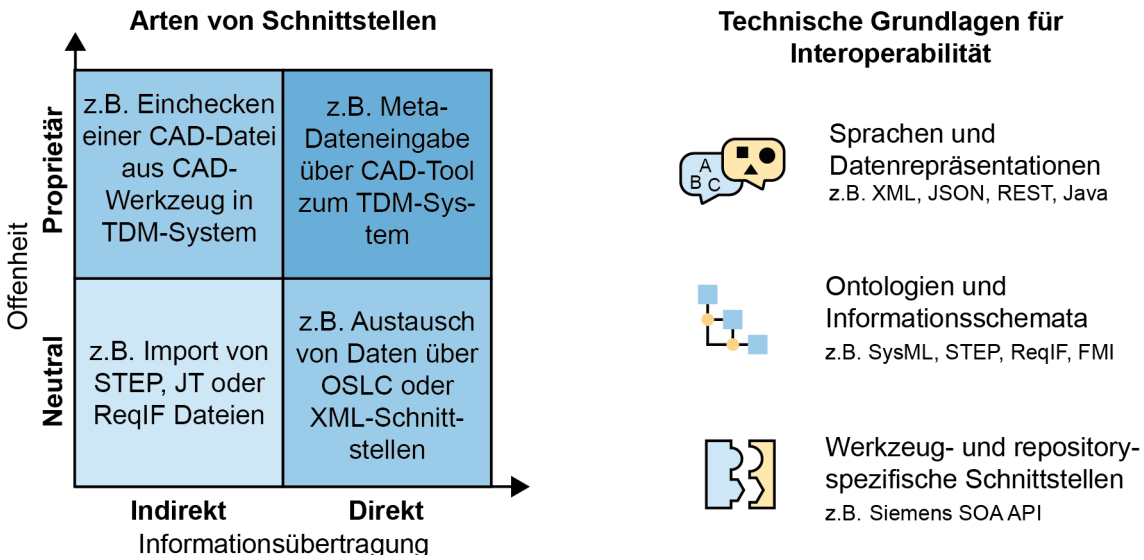


Bild 3-15: Arten von Schnittstellen nach [Sin14, S. 328ff] und dazu genutzte technische Grundlagen nach [FNF+14, S. 222f]

besitzt eine offene Datenformatspezifikation und kann daher applikationsunabhängig implementiert werden [FLL+11, S. 9]. Proprietäre bzw. native Datenformate hingegen lassen lediglich einen Austausch zwischen spezifischen Applikationen zu, welche durch die Softwarehersteller zur Stärkungen der Marktposition nicht offengelegt werden [Sch09, S. 30], [OB18, S. 825]. Die in Kapitel 3.4.3 vorgestellten Werkzeuge nutzen jeweils native Datenformate, wobei zum Austausch jedoch auch neutrale Formate genutzt werden können. Neutrale Datenformate weisen jedoch häufig einen eingeschränkten Funktionsumfang auf [Sin14, S. 333]. Proprietäre Datenformate erfordern hingegen typischerweise eine aufwendige Anpassung der Schnittstellen der betroffenen IT-Werkzeuge [OB18, S. 825].

Zur Umsetzung der Schnittstellen zum Austausch von Artefakten stehen nach FISHER ET AL. drei in Bild 3-15 dargestellte grundlegende **Techniken zur Interoperabilität** bereit [FNF+14, S. 222]. Dazu können Sprachen zur Repräsentation von Daten (z. B. XML [MBK+17, S. 28], JSON [ISO21778, S. 1]), Informationsschemata und Ontologien (z. B. STEP [ISO10303], ReqIF[5] [OMG16b-ol, S. 2], OSLC[6] [SEL13]) genutzt werden. Weiterhin existieren werkzeug- und datenbankspezifische (native) Schnittstellen, sogenannte Application Programming Interfaces (API)[7] (z. B. [Sie10, S. 5]), die stellenweise auf teils neutrale Sprachen und Ontologien zurückgreifen. Die Technologien können je nach Schnittstellentyp und Anforderung ausgewählt werden und legen die Grundlagen für die spätere Implementierung einer Schnittstelle auf physischer Ebene fest.

---

[5] Das Requirements Interchange Format (ReqIF) ist ein offenes Austauschformat für Anforderungen und wird durch die OMG definiert.

[6] Neutrale, direkte Schnittstelle, welche den Zugriff auf Daten anderer IT-Werkzeuge anhand von eindeutigen Bezeichnern, sogenannten Uniform Resource Identifier (URI), ermöglicht [Sin14, S. 331].

[7] APIs sind von IT-Werkzeugen bereitgestellte Programmierschnittstellen, die anderen Applikationen Zugriff auf die Funtionen des IT-Werkzeugs erlauben [Sin14, S. 330].

3.4.6 Treiber zur Veränderung der Engineering-IT

Die bereits skizzierte zunehmende Integration der Engineering-IT-Werkzeuge wird aktuell stark begünstigt. Grund dafür sind Veränderungen im Engineering, die auf mehrere Treiber zurückgeführt werden können. Diese und ihr Einfluss auf die Engineering-IT sollen im Folgenden näher vorgestellt werden und sind in Bild 3-16 zusammengefasst.

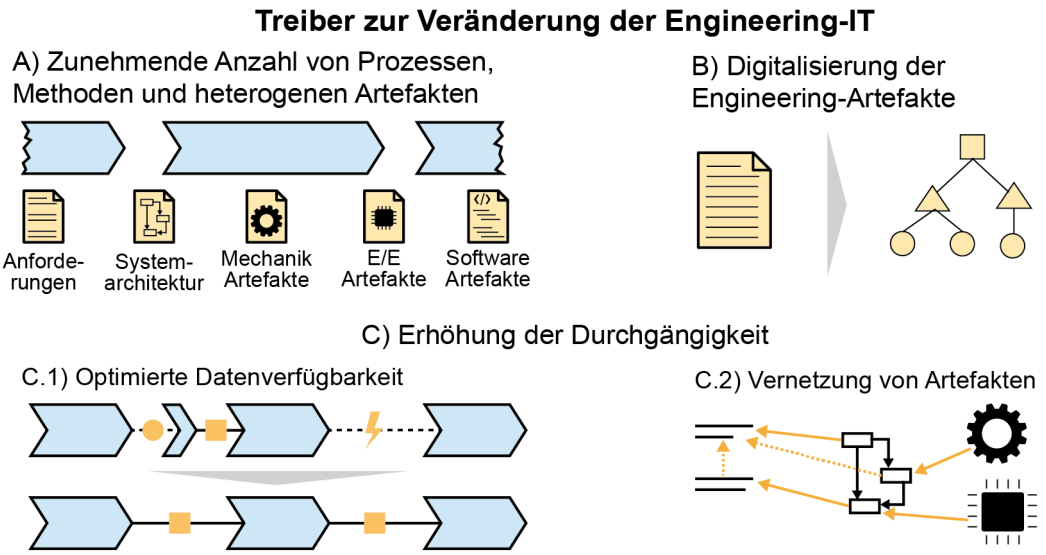


Bild 3-16: Treiber stellen neue Anforderungen an die Engineering-IT und führen zu ihrer Veränderung

Zunehmende Anzahl von Prozessen, Methoden und heterogenen Artefakten: Die Komplexität von ITS schlägt sich im Bedarf für neue Entwicklungsansätze wieder (vgl. Kap. 3.3), bspw. Advanced Systems Engineering, Product Service Engineering oder auch agile Ansätze [DAR+21, S. 28ff], [AH16, S. 22ff]. Diese bringen neue Prozesse und Methoden mit sich, die auf die bestehende, unternehmensspezifische Engineeringlandschaft angepasst und in diese integriert werden müssen. Dies bedingt in der Regel auch veränderte Artefakte. Alle drei Aspekte müssen durch die Engineering-IT berücksichtigt und gezielt unterstützt werden. In den Anfängen mechatronischer Systeme reichten zur Beschreibung der Systeme Stücklisten und CAD-Modelle aus. Im Zeitalter von ITS kommen diese Modelle zu kurz und sind über lineare (z. B. in der Softwaremodellierung) und netzwerkartige Strukturen (z. B. zur Systemmodellierung) zu ergänzen [EMA+16, S. 66].

Ein Beispiel für die Anpassung an die Entwicklung von ITS stellt das Systems Engineering (SE) dar, ein interdisziplinärer Ansatz zur erfolgreichen Entwicklung komplexer technischer Systeme unter Berücksichtigung ihres gesamten Lebenszyklus [WRF+15, S. 265] (s. Kap. 3.4.7). Der Ansatz wird im deutschsprachigen Raum von der Industrie als hoch relevant angesehen und befindet sich aktuell und in den nächsten Jahren in der Implementierung sowie weiteren Entwicklung [DAR+21, S. 67f], [GDS+13, S. 48ff], [SBI+18, S. 17]. Nach der ISO/IEC/IEEE 15288 umfasst das SE ein breites Spektrum heterogener Prozesse. Dies beinhaltet technische Prozesse zur Systemgestaltung, technische Managementprozesse zur Projektabwicklung sowie organisatorische und Vertragsprozesse

[ISO15288, S. 15f]. Dadurch sind nicht mehr länger fokussierte Themeninseln wie die mechanische Konstruktion im Fokus, sondern ein großes Spektrum heterogener Prozessaktivitäten, welches bspw. die Konstruktion als ein Element neben weiteren umfassen kann. Da diese Prozessaktivitäten durch IT-Werkzeuge unterstützt werden, entsteht hier eine Vielzahl von potentieller Anforderungen hinsichtlich notwendiger Funktionalität in IT-Werkzeugen. Damit einher geht der Bedarf, die neuen Artefakttypen, bspw. Modelle zur Systemspezifikation, in den Engineering-IT-Werkzeugen editieren und verwalten zu können.

Die Prozesse der ISO/IEC/IEEE 15288 und ihre Artefakte sind untereinander eng verknüpft [ISO15288, S. 15]. Zudem schließt SE die Nutzung spezifischer Methoden ein [FBK+14, S. 23ff]. Der Einsatz vernetzter Prozesse und Methoden ist zwar für das Engineering besonders effektiv [Gra13, S. 52, 195], kann jedoch zu inkonsistenten und redundanten Bestandteilen von Artefakten führen, wenn sie nicht aufeinander abgestimmt sind. In der Praxis sind Engineering-Methoden häufig nicht integriert und werden isoliert geplant und eingeführt [TG18, S. 11]. So können verschiedene Artefakte aus unterschiedlichen Prozessen und Methoden resultieren, die jedoch den gleichen Sachverhalt beschreiben und in unterschiedlichen IT-Systemen und Datenformaten abgelegt werden [Tie13a, S. 109]. „Funktionen“ können bspw. in Spezifikationen des RE, als Bestandteile funktionaler Systemarchitekturen und als Elemente einer funktionalen gegliederten FMEA[8] beschrieben werden. Bei nicht integrierten Prozessen und Methoden haben diese Artefakte keine Beziehungen zueinander und können in der Praxis inkonsistente Werte enthalten. Abgestimmte Prozesse und Methoden erfordern daher ein tiefes Verständnis der Zusammenhänge [Lin09, S. 61] und eine ganzheitliche Betrachtungsweise, um inkonsistente oder redundante Artefakte zu vermeiden. Dies gilt um so mehr, wenn neben dem SE weitere Ansätze in das Engineering integriert werden sollen.

Digitalisierung der Engineering-Artefakte: Neben einer grundsätzlichen, zunehmenden Anzahl Artefakte wird eine Vielzahl der Artefakte digitalisiert. Digitalisierung kann in diesem Zusammenhang als die Nutzung digitaler Technologien und computer-verarbeitbarer digitaler Daten zur Transformation von Geschäftsprozessen, basierend auf einem digitalen Modell, verstanden werden [WS18, S. 57f], [SW18, S. 6]. Dabei unterstützt das digitale Modell die Leistungserbringung, um die Ergebnisse in der realen Welt sichtbar zu machen [WS18, S. 58]. Die Umsetzung im Engineering wird bereits in Bild 3-7 deutlich, bei dem das digitale Modell des virtuellen PLZ die Informationsprozesse unterstützt.

In der Praxis bedeutet dies neben einer grundlegenden Transformation der betroffenen Prozesse, die Artefakte als Grundvoraussetzung in ein computer-verarbeitbares Format mit einer definierten Struktur und Semantik zu überführen und anzureichern. Heutzutage werden viele Informationen dokumentenbasiert abgelegt, die jedoch hinsichtlich ihrer Struktur und Semantik nicht computer-verarbeitbar sind. Beispiele sind Anforderungen

[8] Failure Mode and Effects Analysis, eine analytische Methode zur sytematischen Erfassung von Fehlern und resultierender Risiken eines Systems [FG13, S. 42].

in nicht-atomar strukturierten, textuellen Dokumenten oder Zeichnungen in Form von semantiklosen Diagrammen. Die fehlende Granularität setzt die manuelle Bearbeitung einzelner atomarer Informationen voraus und hemmt die durch IT-Werkzeuge unterstützte Verwaltung der Informationen, z. B. im Rahmen von Lifecyclemanagement-Prozessen. Eine automatisierte Auswertung, z. B. der Diagrammelemente, ist nicht zuverlässig möglich. Darüber hinaus führen dokumentenbasierte Informationen häufig zu redundanten oder inkonsisteten Informationen, da sie über zahlreiche unsynchronisierte Dokumente verteilt sein können [FMS12, S. 15f], [Alt12, S. 1ff].

Als Beispiel für die voranschreitende Digitalisierung der Artefakte dient u. a. das Model-based Systems Engineering (MBSE), die formalisierte Anwendung von Modellen zur Unterstützung der SE-Prozesse entlang des gesamten PLZ. Durch Nutzung definierter Strukturen sowie Syntax und Semantik sind diese Modelle computer-verarbeitbar und können automatisiert analysiert und zum Teil zu Simulationszwecken ausgeführt werden [Wei14, S. 3, 21ff, 190]. Dank der atomaren, granularen Informationsstrukturierung können Informationen zielgerichteter genutzt und verknüpft werden [FMS12, S. 18f]. Dies bedeutet zum einen aufgrund der Granulatiät jedoch, dass eine Vielzahl neuer Datenobjekte, Attribute und Beziehungen gehandhabt werden muss. Daher müssen detaillierte Informationen zu den Bestandteilen, Strukturen und Beziehungen der Artefakte im Engineering vorliegen, um die Datenstrukturen in der Engineering-IT-Infrastruktur erfassen und abbilden zu können. Zum anderen müssen zusätzliche neue Funktionen von der Engineering-IT-Infrastruktur für die Vielzahl der unterschiedlichen Artefakttypen bereitgestellt werden, z. B. zur Modellbearbeitung oder -verwaltung im Rahmen ihres Lebenszyklus.

Erhöhung der Durchgängigkeit: Das Arbeiten mit Artefakten im Kontext der Digitalisierung schafft vollkommen neue Möglichkeiten. Wie in den vorherigen Kapiteln dargestellt, werden dabei nicht nur erheblich mehr Daten digital bereitgestellt, sondern diese sind auch immer stärker miteinander vernetzt [HK16, S. 979]. Wesentlich ist dabei die Durchgängigkeit und Verfügbarkeit von Daten. Die Durchgängigkeit ist in Bild 3-7 durch die Pfeile angedeutet, ist jedoch auch innerhalb der Prozesse einzelner PLZ-Phasen und zwischen den PLZ-Phasen selbst gefordert. Zur Umsetzung dieser Potentiale gibt es es verschiedene Motivationen, die im Folgenden vorgestellt werden sollen. Wird von Durchgängigkeit gesprochen, so ist damit in diesem Kontext die Vernetzung aller Artefakte über den gesamten PLZ gemeint, ungeachtet ihrer Domänenzugehörigkeit [Eig16, S. 64ff]. Ziel der Durchgängigkeit ist der Austausch von Informationen bzw. Daten zwischen Prozessen ohne oder nur mit geringer manueller Nacharbeit [TOG18, S. 245]. Aus diesen in der Prozessebene definierten Merkmalen leiten sich unmittelbar Anforderungen an die Schnittstellen der IT-Systeme ab, da diese im geforderten Funktionsumfang den Daten- bzw. Informationsaustausch unterstützen müssen. Nachfolgend werden einige praktische Beispiele vorgestellt, welche Anforderungen an die Durchgängigkeit stellen.

Allgemein steht im Trend der Digitalisierung die Transformation der Geschäftsprozesse zu digital unterstützten End-to-End Prozessen. Diese betrachten Prozesse ausgehend vom Bedarf eines internen oder externen Kunden bis zur Leistung an diesen, um dessen Bedarf

abzudecken und damit einen Wert zu erzeugen. Dabei werden alle zur Prozessabwicklung notwendigen Prozessschritte und die darin benötigten Ressourcen betrachtet [Ber12, S. 29f]. In diesem Zusammenhang sind nicht nur Prozesse des Engineering eingebunden, sondern auch diverse Business-Prozesse [HK16, S. 979]. Dabei wird im Sinne der Durchgängigkeit vorausgesetzt, dass bei End-to-End Prozessen mit Beteiligung des Engineerings entsprechende Schnittstellen in die angrenzenden Prozess- und Unternehmensbereiche existieren, um die Daten bzw. Informationen im benötigten Format verarbeiten zu können.

Im Engineering wird die Rückverfolgbarkeit (auch synonym als Traceability bezeichnet) zwischen Artefakten durch normative Vorgaben oder Engineering-Methoden gefordert. Nach der ISO/IEC/IEEE 24765 ist Traceability die Beziehung zwischen zwei oder mehr aus dem Entwicklungsprozess stammenden Artefakten, um die Rückverfolgbarkeit zu inhaltlich vorausgehenden oder nachfolgenden Artefakten zu definieren [ISO24765, S. 481]. Ein Beispiel kann die Rückverfolgbarkeit durch Trace-Links eines Software-Codes auf ein System-Architekturelement sein, welches wiederum eine Anforderung erfüllt. Die DIN EN 61508 fordert zur Sicherstellung der funktionalen Sicherheit eine bidirektionale Rückverfolgbarkeit der Entwicklungsartefakte, über verschiedene Ebenen von Anforderungen, über die Architektur bis zu den Verifikations- und Validierungsmaßnahmen (z. B. zu Testspezifikation oder dem Validierungsplan) [DIN61508, S. 52ff]. Hierzu wird explizit auf die Nutzung von IT-Werkzeugen hingewiesen [DIN61508, S. 84]. Rückverfolgbarkeit wird zudem auch im Rahmen der Komplexitätsbeherrschung über den gesamten PLZ von bestimmten Engineering-Methoden vorausgesetzt [DAR+21, S. 65], [Eig14a, S. 853]. Für die Umsetzung der Traceability in der Engineering-IT sind zum einen entsprechende Funktionalitäten zum Definieren und Verwalten der Trace-Links notwendig, zum anderen Schnittstellen, die es erlauben, zwischen den betroffenen IT-Werkzeugen auf die jeweiligen verknüpften Artefakte zu verweisen. Aufgrund der in den vorherigen Kapiteln dargelegten Umstände müssen dabei ggf. zahlreiche heterogene Artefakttypen in verschiedenen IT-Werkzeugen verknüpft werden.

Die Systemkomplexität setzt insbesondere aufgrund ihrer Varianz eine konsistente Verwaltung einer Vielzahl von Artefakten über ihren Lebenszyklus voraus. Dabei nimmt insbesondere das Konfigurationsmanagement (Configuration Management, CM) eine bedeutende Rolle ein. CM wird bspw. durch die ISO/IEC/IEEE 15288 zur effektiven Durchführung der Systementwicklung gefordert [ISO15288, S. 39ff]. Zudem nimmt die Varianz von Produkten zu (s. Kap. 3.2), welche ebenfalls mittels CM abgebildet und gehandhabt werden kann [Ger16, S. 230].

Konfigurationsmanagegement ist nach ISO/IEC/IEEE 24765 eine Managementtätigkeit, bei der technische und organisatorische Regeln auf Konfigurationseinheiten (Configuration Items, CI) über den gesamten PLZ angewendet werden. Eine Konfiguration besteht aus ihr zugeordneten Konfigurationseinheiten, welche konsistent miteinander verbundene funktionelle und physische Merkmale eines Produktes repräsentieren [ISO24765, S. 90f]. Beispielsweise kann durch das CM definiert werden, welche Software-Releases mit einer definierten Revision eines Steuergerätes unter Berücksichtigung dessen Kontexts

kompatibel sind. Dies setzt voraus, dass alle Artefakte, die entlang des PLZ entstehen und als CI definiert werden, unter eine konsistente Konfigurationskontrolle gestellt werden könnnen. Dazu müssen sämtliche IT-Werkzeuge, die CI's verwalten, entsprechende Funktionen zur Konfigurationskontrolle der Artefakte bereitstellen und über Schnittstellen die Integrität der gesamten Konfiguration eines Systems sicherstellen können [FNF+14, S. 218]. Darüber hinaus sind in der Praxis weitere Lebenszyklusmanagementprozesse zu finden, bspw. Varianten-, Versions- oder Änderungsmanagement. Aufgrund abweichender Prozessverständnisse oder historischer Prozessgrenzen können dabei mehrere Lebenszyklusmanagementprozesse existieren, die zur effektiveren Prozessausübung konsolidiert werden sollten.

Für die Synchronisierung zwischen virtuellen und physischen PLZ und Abdeckung des gesamten PLZ ergeben sich neue Perspektiven. So bieten heutige PLM-Umgebungen oder ERP-Systeme fast ausschließlich Funktionen, um den PLZ bis zum Ende der Produktion zu verwalten. Dabei besteht ein enormes Potential, die Produktinbetriebnahme und -nutzung sowie End of Life Phase mit einzubeziehen. Dies spiegelt sich im Konzept des digitalen Zwillings wider [BR16, S. 60ff]. Dies setzt zum einen voraus, dass ein produktinstanzspezifisches Artefakt mit für den Nutzungszweck notwendigen Inhalten besteht, was sich über eine eindeutig zu identifizierende Konfiguration (z. B. mittels Seriennummer) verwalten lässt [Eig16, S. 64ff]. Die Konfiguration des virtuellen Abbilds muss dabei immer dem realen Abbild entsprechen und bei Modifikationen oder Service-Maßnahmen entsprechend nachgepflegt werden. Zum anderen muss die Datenrückführung vom Produkt in die Engineering-IT-Infrastruktur über eine abgestimmte Schnittstelle möglich sein und entsprechende Funktionen zur Nutzung und Verwaltung der Daten gegeben sein.

Durchgängigkeit ermöglicht durch die automatisierte Datenverarbeitung vollkommen neue Anwendungsfelder. Durch die Vernetzung von Maschinen und Systemen über geeignete Schnittstellen können Informationen ausgetauscht, ausgewertet und im Rahmen definierter Grenzen auch selbstständige Entscheidungen getroffen werden [BHS15, S. 70]. So können bspw. die Betriebsdaten einer Maschine mit den Simulationsdaten des virtuellen Abbilds im Rahmen eines Regelkreises abgeglichen werden und das virtuelle Systemmodell weiter optimiert werden [BR16, S. 72], [AAP+19, S. 5ff]. Die manuelle Aufbereitung und der Abgleich der Daten wären zu zeitaufwendig. Diese Möglichkeit nutzt auch das Prinzip der Engineering Intelligence, bei dem vernetzte Engineering-Artefakte zur Informationsgewinnung ausgewertet werden. Als Grundlage wird die graphenbasierte Modellierung genutzt und um Wissenstransformation bei einer entsprechend gegebenen Datenintegration ergänzt [HK16, S. 984]. Die Unterstützung durch künstliche Intelligenz ist denkbar. Dadurch können bspw. Produktinformationen automatisiert aggregiert werden, z. B. um Produktvarianten zu generieren [its19]. Bei entsprechender Durchgängigkeit ist somit teilweise oder vollständige Automatisierung von Prozessschritten denkbar [Gad17, S. 11, 21].

3.4.7 Systemische Betrachtung der Engineering-IT

Sowohl die technischen Systeme als auch das Engineering selbst werden aufgrund der dargelegten Umstände immer komplexer. Ein geeigneter und bereits erwähnter Ansatz zur Komplexitätshandhabung ist das Systems Engineering. Dieser erlaubt eine systemische Betrachtung der Engineering-IT und soll nachfolgend eingeführt werden.

Ein **System** wird nach der ISO/IEC/IEEE 15288 definiert als „combination of interacting elements organized to achieve one or more stated purposes“ [ISO15288, S. 9]. Systeme können aufgrund der breiten Definition sehr vielfältiger Natur sein. Nach der Definition des International Council on Systems Engineering (INCOSE) können Systeme aus Produkten, aus Hard- und Software, Prozessen, Personen, Informationen und weiteren unterstützenden Elementen bestehen [WRF+15, S. 5]. Somit können sowohl ein Auto, das Verkehrssystem, in dem das Fahrzeug später genutzt wird, und auch die Engineering-Organisation, in der es entwickelt wird, jeweils ein System darstellen.

Nach ROPOHL gibt es drei Aspekte, die ein System umfasst, welche die Definition eines Systems erläutern und vertiefen [Rop09, S. 75ff]. Diese sind nachfolgend dargestellt und in Bild 3-17 zusammengefasst.

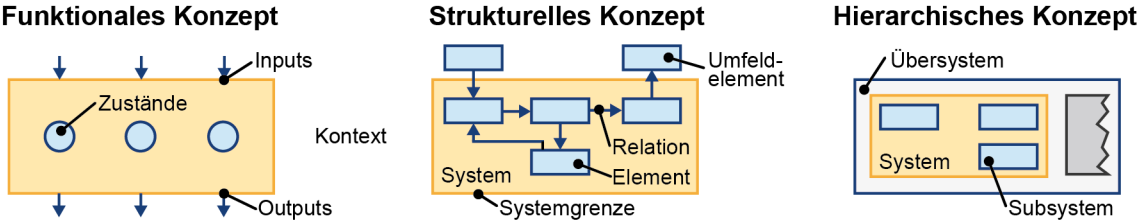


Bild 3-17: Drei Konzepte zur Betrachtung von Systemen in Anlehnung an [Rop09, S. 76]

Aus *funktionaler Sicht* erfüllt das System eine Funktion, indem es einen Input in einen Output transformiert. Dabei kann das System verschiedene Zustände einnehmen. Es steht die Frage im Vordergrund, *was* ein System tut. Die Frage, *wie* es seine Funktion erfüllt und welche Struktur hierfür notwendig ist, wird explizit nicht betrachtet. Daher wird diese Betrachtungsweise auch als Black-Box bezeichnet, da der innere Aufbau des Systems an dieser Stelle vollkommen unerheblich ist [Rop09, S. 75f], [HWF+18, S. 36f].

Aus *struktureller Sicht* setzen sich Systeme aus Systemelementen zusammen. Dabei ist zu beobachten, dass Systeme Fähigkeiten und Eigenschaften besitzen, welche die Summe der isoliert betrachteten Systemelemente nicht aufweisen. Dieser Effekt wird auch als Emergenz bezeichnet [Hit07, S. 7]. Die Emergenz ergibt sich durch die Relationen zwischen den einzelnen Systemelementen, welche unterschiedliche Systemeigenschaften hervorrufen können. Es ist jedoch auch die Beschaffenheit der Elemente zu berücksichtigen, da diese maßgeblich definiert, wie gut sich ein Element in ein System integrieren lässt und zur Emergenz beitragen kann [Rop09, S. 75]. Daher muss ein System immer in seinen Kontext gesetzt werden und mit diesem ganzheitlich betrachtet werden [Pat82, S. 6ff]. Ganzheitlichkeit berücksichtigt demnach emergente Effekte und verzichtet somit auf eine isolierte Betrachtung ohne die Berücksichtigung der Relationen eines Systems. Um die

Funktionsweise eines Systems ergründen zu können, ist das Verständnis dieser Relationen essentiell [Hit07, S. 21ff]. Diese Betrachtungsweise wird auch als White-Box bezeichnet [HWF+18, S. 33].

Der Kontext wird durch die Systemgrenze vom System abgegrenzt. Die Definition der Systemgrenze ist abhängig von der Problemstellung und unterliegt somit einer gewissen Willkür [UP88, S. 50]. Es ist aber zu berücksichtigen, dass innerhalb der Systemgrenze ein stärkeres und relevanteres Maß an Beziehungen vorliegt [Har49, S. 332]. Systeme können Beziehungen zu Elementen im Umfeld aufweisen. Ein solches System wird als offenes System bezeichnet, die Wechselwirkungen erlauben eine Beeinflussung des Systems durch die Umwelt und gleichzeitig eine Beeinflussung der Umwelt durch das System [UP88, S. 50]. Systeme ohne Beziehungen zur Umwelt werden als geschlossene Systeme bezeichnet, existieren jedoch in der Praxis nicht und somit nur als vereinfachendes, theoretisches Konstrukt [BFP96, S. 9].

Da Systeme in ihrem Kontext zu sehen sind, kann der Kontext aus *hierarchischer Sicht* selbst ein System darstellen, das System ist somit Teil eines größeren, übergeordneten Systems. Dieses wird auch als Über- oder Supersystem bezeichnet. Dadurch ergibt sich der Umstand, dass System-Elemente wiederum selbst Systeme darstellen können, die dann Subsysteme des anfänglich betrachteten Systems darstellen [HWF+18, S. 30]. Somit bilden sich Hierarchiestufen, bei denen ein Abwärtsbewegen eine detaillierte Erklärung des Systems hervorbringt, ein Aufwärtsbewegen hingegen ein besseres Verständnis für dessen Bedeutung im Gesamtkontext [Rop09, S. 77]. Damit kann je nach Problemstellung die Abstraktionsebene angepasst werden.

Das INCOSE beschreibt **Systems Engineering** als einen interdisziplinären Ansatz zur erfolgreichen Umsetzung von technischen Systemen. Es hat als Ziel, alle beteiligten Domänen und spezifischen Sichtweisen auf das System zu integrieren [WRF+15, S. 11]. Dabei wird auf das Systemdenken zurückgegriffen, um Systeme ganzheitlich verstehen und gestalten zu können [HWF+18, S. 30]. SE ist ein Ansatz, der sämtliche Lebenszyklusphasen des Systems von der frühen Konzeptphase bis zum End-of-Life mit einbezieht (vgl. Bild 3-7) [WRF+15, S. 12, 25]. HABERFELLNER ET AL. sehen vier Grundprinzipien im SE als essentiell an. Dazu gehört, Systeme vom Groben zum Detail zu gestalten, das Denken in verschiedenen Lösungsalternativen, die Gliederung der Systementwicklung nach zeitlichen Gesichtspunkten (Phasenablauf) und bei der Problemlösung einem formalen Vorgehensleitfaden zu folgen (Problemlösungszyklus) [HWF+18, S. 53].

Auch Organisationen oder ihre Teile können Systeme darstellen, z. B. die Engineering-IT. Das **Enterprise Systems Engineering** (ESE) nutzt daher Systems Engineering zur Analyse, Spezifikation, Gestaltung und Umsetzung von Organisationen [SCC+09, S. 497]. Organisationen werden dabei als sozio-technische Systeme, bestehend aus Menschen, Technologien und Informationen, betrachtet [Sno14, S. 11]. Wesentlich für eine erfolgreiche Transformation von Organisationen ist dabei die ganzheitliche, systemische Betrachtung, insbesondere ihrer internen und externen Relationen [MFL19]. Gründe für eine notwen-

dige Transformation des Systems „Organisation“ können aus internen Faktoren oder der Relationen zu dessen Umfeld resultieren, z. B. aufgrund sich verändernder Produkte oder Marktbedingungen.

Um die Komplexität einer Organisation handhabbar zu machen, bedient man sich konsequenterweise auch der Modellierung. Das Modell einer Organisation beschreibt die Enterprise Architektur (EA). Dabei existieren verschiedene Ansätze zu ihrer Modellierung [Sno14, S. 5f]. Während diese in der Vergangenheit stark IT-zentriert waren, ist verstärkt eine ganzheitlichere Sicht auf die Organisation wahrzunehmen [Sae05, S. 7f].

3.4.8 Gestaltung der Engineering-IT-Infrastruktur

Die vorgestellten Treiber zur Veränderung der Engineering-IT lassen die bisherigen IT-Infrastrukturen an ihre Grenzen stoßen. Stattdessen ist laut EIGNER ET AL. ein „totales Um-, Quer- und Neudenken“ notwendig [Eig16, S. 64]. Die unter diesen Gesichtspunkten erforderlichen Aufgaben zur Gestaltung der Engineering-IT sollen daher im Folgenden näher erläutert werden.

Oftmals werden IT-Infrastrukturen im Rahmen eines **IT-Management-Ansatzes** gestaltet. Dieser gibt auch den Handlungsrahmen für die in Kapitel 3.4.1 beschriebenen Aufgaben der Engineering-IT vor. HANSCHKE ET AL. sehen folgende Ziele [HGG16, S. 9]:

- **Verständnis schaffen:** Strukturen und Zusammenhänge auf fachlicher und organisatorischer Ebene und den Änderungsbedarf und dessen Auswirkungen verstehen.
- **Gestaltung ermöglichen:** Geeignete fachliche Lösungen für die strategischen bzw. operativen Geschäftsanforderungen gestalten.
- **Umsetzung steuern:** Die fachliche Lösung umsetzen und dabei effektiv steuern.
- **Effektiv kommunizieren:** Die Geschäftsanforderungen und möglichen Lösungsansätze müssen für alle Beteiligten effektiv und verständlich kommuniziert werden.

Die Umsetzung dieser Ziele geschieht häufig im Rahmen des **Enterprise Architecture Managements** (EAM). Nach HANSCHKE ET AL. ist EAM ein systematischer und ganzheitlicher Ansatz, um das Verständnis, die Kommunikation, das Gestalten und Planen der fachlichen und technischen Strukturen eines Unternehmens zu unterstützen. Dabei hilft EAM, die Komplexität der IT-Landschaft zu beherrschen und sie bedarfsorientiert weiterzuentwickeln [HGG16, S. 314]. Dazu integriert das EAM die wesentlichen Strukturen der Organisation in einem Modell und setzt sie miteinander in Beziehung [HGG16, S. 15]. Je nach Definition zielt EAM dabei auf das unternehmensweite IT-Management bis hin zu einer gezielten Transformation ganzer Organisationen ab [WLF14, S. 1]. Etablierte EAM-Frameworks wie TOGAF (The Open Group Architecture Framework) oder EAP (Enterprise Architecture Planning) nutzen dazu die vier in Tabelle 3-2 dargestellten Betrachtungsebenen zur Beschreibung der Architektur [Sch17, S. 19f], [Tie13a, S. 102ff].

Tabelle 3-2: Grundlegende Architekturebenen in EAM-Frameworks und Einordnung der Bestandteile des Engineerings

Architektur-ebene	Beschreibung	Beispiel
Geschäfts-architektur	Präzisierung der Geschäftsstrategie durch Gestaltung der Geschäftsprozesse; ggf. auch Organisationsstrukturen und Ressourcen	Engineering *Prozesse* und *Methoden* eines Maschinen-baubetriebs
Daten-architektur	Elemente, Strukturen und Beziehungen der logischen und physischen Datenbestände	*Artefakte* des Engineerings mit Syntax und Semantik
Applikations-architektur	Struktur der IT-Werkzeuge und ihrer Schnitt-stellen; IT-Werkzeuge stellen eine Funktiona-lität zur Unterstützung der Geschäftsprozesse bereit und nutzen dabei Daten	*IT-Werkzeuge* und ihre Schnittstellen aus fachlicher Perspektive
Infrastruktur-architektur	Struktur von Technologie-Komponenten wie Hard- oder Software, um die Anwendungen zu unterstützen	Basisinfrastruktur, z.B. Daten-bankmanagementsysteme

Geschäftsarchitektur: Die Geschäftsarchitektur beschreibt die fachlichen Strukturen der Organisation, bspw. das Prozessmanagement oder die Organisationsentwicklung. Sie bilden den Ordnungs- und Bezugsrahmen für die Planung und Steuerung der IT. Ein zentrales Element zur Beschreibung der Geschäftsarchitektur sind Prozessmodelle auf einer Granularitätsebene, die eine ganzheitliche Sicht erlaubt [HGG16, S. 15f]. Neben den Prozessen können auch die Organisationsstrukturen, Ressourcen wie Akteure oder Sachmittel sowie Geschäftsziele mit einbezogen werden. Die Geschäftsarchitektur kann somit die relevanten Geschäftsfelder, Dienstleistungen und die dafür benötigten Funktionen, Prozesse, Standorte, Rollen und Informationen beschreiben [Tie13a, S. 107f]. Wesentliche Informationsquelle für die Geschäftsarchitektur sind die dargestellten Fachbereiche, weshalb diese gemeinsam mit der IT an deren Erstellung und Pflege arbeiten sollten (s. Kap. 3.4.2) [HGG16, S. 16].

Datenarchitektur: Die Datenarchitektur dient der Identifizierung und Dokumentation der von den Geschäftsprozessen benötigten Artefakte. Sie beschreibt die benötigten Datentypen und ist somit die Grundlage für die spätere Auswahl von IT-Werkzeugen. Darüber hinaus werden die jeweiligen Datenentitäten, ihre Attribute und Beziehungen beschrieben [SH93, S. 169]. Ziel ist es, ein ganzheitliches und integrierendes Modell der Daten zu generieren. Dadurch können Daten-Redundanzen vermieden werden. Zudem können die Daten mit den jeweiligen Geschäftsprozessen verknüpft werden. Die Datenarchitektur kann sowohl logische als auch physische Datenbestände beschreiben. Typischerweise werden für die Erstellung von Datenarchitekturen die bestehenden Anwendungen und Informationsquellen analysiert, um alle Datenobjekte zu erfassen [Tie13a, S. 108f].

Applikationsarchitektur: Die Applikationsarchitektur definiert die zur Unterstützung der Geschäftsprozesse und zur Bearbeitung bzw. Verwaltung der Daten notwendigen IT-Werkzeuge [SH93, S. 199]. Sie gibt eine Übersicht über die IT-Werkzeuge und erlaubt

die Analyse und Dokumentation ihrer Interaktionen untereinander mittels Schnittstellen. Die Applikationen können mit den unterstützten Geschäftsprozessen und Technologien verknüpft werden [HGG16, S. 16]. Die Applikationsarchitektur fungiert somit als Spezifikation und macht konkrete Vorgaben für die Ausgestaltung und Entwicklung. Typischerweise werden die IT-Anwendungen nach den Hauptprozessen oder Funktionen gegliedert. Dabei kann sowohl eine funktionale Sicht zur Beschreibung der Prozessunterstützung als auch eine technische Sicht zur Beschreibung der IT-Systeme und ihres Zusammenwirkens eingenommen werden [Tie13a, S. 106f].

Technologiearchitektur: Die Technologiearchitektur definiert die notwendigen Technologien und Konzepte, um eine Betriebsumgebung für die Applikationen bereitzustellen [SH93, S. 223]. Damit verbindet sie die fachliche, strategische mit der operativen Welt. Sie trägt dazu bei, technologische Standards zu definieren, z. B. welche Netzwerk- oder Datenbankmanagementsysteme zu nutzen sind [HGG16, S. 19]. Die Technologiearchitektur kann dabei Hard- und Software beschreiben, z. B. Clients, Server-, Storage-, Backupsysteme, IT-Netze bis hin zu Peripheriegeräten [Tie13a, S. 105]. Hinsichtlich der Software können verschiedene Schichten differenziert werden, um z. B. Teile einer Funktion durch verschiedene IT-Werkzeuge auf verschiedenen Schichten umzusetzen. Die Präsentationsschicht stellt die Benutzerschnittstelle bereit und dient der Daten-Repräsentation und Aufnahme von Benutzereingaben, z. B. mittels einer graphischen Benutzeroberfläche. Die Logikschicht beinhaltet die Anwendungslogik zur Verarbeitung der Daten und Nutzerbefehle (z. B. PDM-Kernfunktionen). Die Datenhaltungsschicht umfasst das Speichern und Bereitstellen von Daten (z. B. mittels einer relationalen Datenbank) [GR96, S. 10f], [Tie13a, S. 105f], [VWZ+18, S. 522].

Das EAM unterstützt somit ein systematisches Vorgehen zur Gestaltung der Engineering-IT, welches zwingend notwendig ist [Tie13b, S. 23]. Architekturen sind die Grundlage für einen planvollen Abgleich der IT-Ebene mit der Prozessebene („Strategic Alignment“) und erlauben eine Standardisierung, die bspw. zu höherer Datenverfügbarkeit führen kann [Tie13a, S. 92]. Im Gegensatz dazu stehen Ad-hoc Entscheidungen, wie sie aktuell häufig in der Engineering-IT getroffen werden [AZL19, S. 9]. Es fehlt eine ganzheitliche Sicht auf die fachlichen Strukturen der Prozessebene, die Anwender und IT-Werkzeuge. Dadurch werden einzelne Entscheidungen getroffen, die keiner übergreifenden Strategie folgen und bei denen keine Gesamtsicht über alle Aufgaben im Engineering vorliegt [Tie13a, S. 91].

Zusammenfassend wird bei einer Enterprise Architektur eine Organisation oder Teile von ihr als System betrachtet, wobei TOGAF den Begriff der Architektur definiert als [TOG18, S. 6, 11]:

> *„The structure of components, their inter-relationships, and the principles and guidelines governing their design and evolution over time“*

Mit Hinblick auf die in Kapitel 3.3 beschriebenen Bestandteile der Engineering-IT lassen sich aus den generischen Architekturebenen folgende Kernaufgaben zur Gestaltung der Engineering-IT ableiten.

Identifikation und Analyse der Anforderungen: Zunächst muss eine Geschäftsarchitektur des Engineerings abgebildet werden, welche die Prozessebene vollständig umfasst. Dazu gehören im Engineering die Prozesse, Methoden und die dazugehörige Beschreibung der Artefaktbestandteile und -beziehungen, die in Form entsprechender Architekturen abgebildet werden können. Für die Informationsgewinnung und Modellierung müssen alle relevanten Stakeholder mit einbezogen werden. Anhand dieser Architekturen ergeben sich Anforderungen aus der Prozessebene an die IT. Typischerweise müssen die Anforderungen strukturiert verwaltet werden [Han16, S. 132ff, 185]. Aus den Anforderungen ergeben sich spätere Themenbündel und notwendige Funktionalitäten für die Anwendungsarchitektur.

Definition einer Applikationsarchitektur: Um die Brücke zwischen Prozess- und IT-Systemebene zu schlagen, existieren verschiedene Ansätze, z. B. die Definition von Services, Funktionen oder logischen Entitäten. Sie sind Hilfsmittel, um verschiedene Abstraktionsebenen der IT-Werkzeuge abzubilden. So können auf einer logischen Ebene die Arten von Applikationen verortet werden, wohingegen auf physischer Ebene eine tatsächlich implementierbare Applikation verortet werden muss [TOG18, S. 298]. Letztlich muss zur Umsetzung jedoch eine physische Applikationsarchitektur existieren. Die Applikationsarchitektur beschreibt neben den genutzten logischen bzw. physischen Entitäten auch ihre Beziehungen untereinander, die sich zur Umsetzung der Anforderungen aus der Prozessebene ergeben.

Auswahl implementierbarer Applikationen: Die Auswahl physischer Applikationen ist abhängig vom Erfüllungsgrad der Anforderungen. Dabei existiert typischerweise ein Entscheidungsproblem, da verschiedene oder sich gegenseitig ausschließende IT-Werkzeuge ausgewählt werden können. Jedes IT-Werkzeug weist dabei unterschiedliche Charakteristika auf und deckt unterschiedliche Anforderungen ab. Daher werden typischerweise Kriterienkataloge mit Anforderungen genutzt, um die einzelnen Werkzeuge zu bewerten und auszuwählen. Die Bewertungskriterien variieren dabei je nach Bewertungsansatz und sind situationsabhängig [SV11, S. 88ff], [Lei15, S. 248ff], [Lan09, S. 206ff]. Im Kern ergeben sich nach der Struktur von SCHÜTTE UND VERING vier Kriterienkategorien, die sich im Wesentlichen in den vorgestellten Ansätzen übereinstimmend wiederfinden. Zunächst sind dies die fachliche Unterstützung (Funktionalität, Interoperabilität, Unterstützung der Prozessebene), qualitative, vor allem benutzerbezogene Kriterien (Gebrauchstauglichkeit, Performanz)[9], betriebswirtschaftliche Kriterien (Anschaffung, Wartung, Anpassung) und das Risiko (Größe des Softwareanbieters, Sichergestellte Wartung und Weiterentwicklung). Darüber hinaus existieren Eigenschaften hinsichtlich der Integration in Hard- und Softwarearchitekturen, wobei moderne Standardsoftwaresysteme diese Eigenschaften typischerweise aufweisen [SV11, S. 21], [Lei15, S. 248ff], [Lan09, S. 206]. Die Technologiearchitektur wird daher im Folgenden nicht näher betrachtet.

[9] Im Folgenden werden insbesondere die funktionalen und qualitativen Kriterien bzw. Anforderungen wieder aufgegriffen.

Es können sich Wechselwirkungen zwischen der Applikationsauswahl und der Definition der Applikationsarchitektur ergeben, da die Auswahl einer einzelnen Applikation Änderungen an weiteren Stellen der Applikationsarchitektur nach sich ziehen kann. Daher kann die Auswahl der Software nicht als Einzelproblem aufgefasst werden und muss als Bestandteil eines übergeordneten Problems zur Gestaltung der Engineering-IT betrachtet werden [SV11, S. 52]. Trotz dieser Feststellung weisen die drei in diesem Absatz beispielhaft zitierten Auswahlansätze für Standardsoftware keine explizite Verknüpfung zum Architekturprozess auf.

Zusammenfassend stellt die Engineering-IT die Organisation und Ansätze zur Gestaltung der IT-Systeme für das Engineering bereit. Dabei sind verschiedenste Stakeholder involviert, die alle das gleiche Verständnis der Problemstellung und des geplanten Lösungskonzeptes benötigen. Die Engineering-IT betreibt eine Vielzahl heterogener IT-Werkzeuge, die unterschiedliche Integrations-Möglichkeiten aufweisen. Verschiedene Treiber verändern die Anforderungen an die IT-Werkzeuge jedoch massiv. Es gilt, neue Prozesse und Methoden sowie eine Vielzahl heterogener Artefakte zu unterstützen. Dabei wird aufgrund der neuen Entwicklungsansätze und -paradigmen eine hohe Durchgängigkeit und Vernetzung der Daten und Informationen im Engineering gefordert. Bestehende IT-Architekturen können diesen Anforderungen nicht gerecht werden, perspektivisch steigt die Menge und Integrationstiefe der IT-Werkzeuge. Die Ad-hoc Integration von IT-Werkzeugen stößt an ihre Grenzen, stattdessen ist eine ganzheitliche Betrachtung notwendig. Die Engineering-IT steht vor der Herausforderung, insbesondere die aufeinander aufbauenden Geschäfts-, Daten- und Applikationsarchitekturen zu analysieren und neu zu gestalten.

3.5 Problemabgrenzung

Die Entwicklung intelligenter technischer Systeme bedarf der verzahnten Zusammenarbeit vieler unterschiedlicher Disziplinen. Grundlage für ihre Arbeit bilden die digitalen Artefakte des Engineerings, welche alle relevanten Aspekte eines technischen Systems beschreiben. Das heutige Engineering befindet sich im Wandel, um mit der Entwicklung der technischen Systeme Schritt halten zu können. Das Engineering nutzt auf der Prozessebene Prozesse und Methoden, in denen In- und Outputs als Artefakte genutzt werden. Der Prozessebene steht die IT-Systemebene gegenüber, welche die IT-Werkzeuge zur Unterstützung der Prozessebene umfasst. Dabei gibt es aktuell durch die Implementierung neuer Entwicklungsansätze wie des SEs zahlreiche Änderungen auf der Prozessebene. Bestehende IT-Infrastrukturen werden diesen Anforderungen in der Regel nicht mehr gerecht und müssen grundlegend umgestaltet werden. Eine Ad-hoc Planung und Integration sowohl auf Prozess- als auch IT-Systemebene sind hinsichtlich der Komplexitätstreiber (vgl. Kap. 3.4.6) jedoch nicht mehr ausreichend. Ein systematischer und ganzheitlicher Ansatz zur Gestaltung der Engineering-IT bringt daher wesentliche **Nutzenpotentiale** mit sich.

- **Integriertes Prozess- und Methodennetzwerk:** Prozesse und Methoden sind aufeinander abgestimmt und nutzen die gleichen, redundanzfreien Bestandteile der Modelle im Engineering. Die ganzheitliche Sicht erlaubt, Schnittstellen zwischen den Prozessen und Methoden über ihre In- und Outputs aufzuzeigen, um die Bildung eines effektiven Netzwerkes zu gestalten. Die inselartige Einführung und Gestaltung von Prozess- und Methodenaspekten wie bspw. Kostenmanagement als nicht integriertes Querschnittsthema [WRF+15, S. 211ff] wird vermieden. Stattdessen können die Aspekte mit bestehenden Prozessen und Methoden verknüpft oder in diese integriert werden. Redundante Prozesse, z. B. aus verschiedenen Disziplinen, können harmonisiert und zur effektiveren Prozessdurchführung integriert werden.
- **Effektive Unterstützung der Nutzer:** Die IT-Werkzeuge weisen die aus fachlicher Sicht notwendigen Funktionen und Eigenschaften auf, um die Aufgaben aus der Prozessebene effektiv zu unterstützen. Dies kann beispielsweise der Bedarf für Funktionen zur Auswertung bestimmter Artefakttypen sein. Aus Sicht der Nutzer werden Werkzeuge mit einer möglichst guten Gebrauchstauglichkeit bereitgestellt, um ihre Aufgaben effizient erledigen zu können.
- **Höhere Datenverfügbarkeit und Durchgängigkeit:** Aufgrund der gezielten Vernetzung der Informationen durch entsprechend funktionale und kompatible Schnittstellen zwischen den IT-Werkzeugen können umfangreiche Nutzenpotentiale umgesetzt werden. So kann eine Vor- und Rückverfolgbarkeit (Traceability) zwischen den Artefakten sichergestellt werden. Informationen können zwischen verschiedenen Prozess- oder Methodenbestandteilen nahtlos bereitgestellt werden, ohne dass sie manuell konvertiert werden müssen. Darüber hinaus können übergreifende Lebenszyklus-Managementprozesse wie das Konfigurationsmanagement die konsistente Steuerung der Artefakte über ihren Lebenszyklus sicherstellen. Dabei werden alle relevanten Artefakttypen berücksichtigt. Dies stellt sicher, dass stets eine vollständige, konsistente und aktuelle Beschreibung des technischen Systems existiert.
- **Bessere Wartbarkeit der IT-Werkzeuge:** Die integrierende Sicht unterstützt die Engineering-IT beim Betrieb ihrer Systeme und bei der Reaktion auf Änderungen. So kann der Änderungsaufwand bei Änderungen in Prozessen oder Methoden schneller abgeschätzt und diese einfacher implementiert werden. Hohe Kosten durch teures Customizing und laufende Wartungskosten von Schnittstellen können durch eine strategisch günstige Auswahl der IT-Werkzeuge reduziert werden. IT-Werkzeuge unterliegen selbst einem Lebenszyklus [Krc15, S. 173ff] und können mit Hilfe geeigneter Gestaltungstechniken gezielt angepasst, erweitert oder ausgetauscht werden.

Ein Gestaltungsansatz für die Engineering-IT muss alle relevanten Bestandteile des Engineerings berücksichtigen. Als Planungsgrundlage müssen Modelle zunächst die fachliche Sicht auf Prozesse und Methoden beschreiben (Geschäftsarchitektur). Aus diesen leitet sich dann eine Architektur der Artefakte im Engineering ab (Datenarchitektur). Aus der Prozessebene ergeben sich somit Anforderungen an IT-Werkzeuge und Schnittstellen hin-

sichtlich der Unterstützung der Prozess- und Anwenderaktivitäten, welche identifiziert und als Bewertungskriterien für die spätere IT-Architektur dokumentiert werden müssen. Mit diesem Wissen kann die Applikationsarchitektur für das Engineering entwickelt werden, welche die IT-Werkzeuge und ihre Schnittstellen beschreibt. Ziel ist dabei die bestmögliche Kombination der Werkzeuge zur Abdeckung der funktionalen und qualitativen Anforderungen und ressourcenarm umsetzbaren Schnittstellen. Die Zielerreichung einer Architektur muss gegenüber den Bewertungskriterien evaluiert werden. Die Erstellung und Modellierung der entsprechenden Modellinhalte, die letztlich in einer IT-Architektur münden, ist komplex, was den folgenden **Herausforderungen** geschuldet ist.

- **Komplexität der Prozessebene:** Im Engineering fehlt es häufig an einem integrierten Modell, welches Prozesse, Methoden und die resultierenden Artefakte und ihre Beziehungen untereinander darstellt. EAM-Frameworks bieten zwar Modelle zur Modellierung der Prozesse und Daten, Methoden können jedoch nicht explizit abgebildet werden und sie zeigen nicht auf, wie die Modellelemente methodisch identifiziert und abgeleitet werden können. Besonders erschwerend ist dabei, dass Engineering Prozesse als besonders komplex betrachtet werden, da sie ein eng vermaschtes Netzwerk von Informationsflüssen darstellen. Die Prozess- und Methodennetzwerke stellen hier (neben dem Nutzenpotential) auch eine Herausforderung dar, da sie die Komplexität erhöhen. Die methodische Unterstützung zur Analyse der Prozessebene ist notwendig, um ein möglichst zutreffendes Abbild des gewünschten Zielzustandes zu modellieren. Darüber hinaus existieren häufig keine ganzheitlichen Datenmodelle über alle Prozesse und Methoden hinweg. Dies führt zu redundanten Daten, die über die verschiedenen Prozesse und Methoden nur mit tiefgreifendem Verständnis zu identifizieren und voneinander abzugrenzen sind. Die fachlichen Beziehungen zwischen den Artefakten bleiben dabei oft unklar.
- **Vernetzung und Einschränkungen der IT-Werkzeuge:** In der Praxis weisen IT-Werkzeuge unterschiedliche Profile im Hinblick auf ihre Eigenschaften auf. Sie besitzen unterschiedliche Stärken und Schwächen hinsichtlich der Erfüllung funktionaler und qualitativer Anforderungen. Daher sind unterschiedlichste Architekturen mit verschiedenen Kombinationen von IT-Werkzeugen denkbar. So lässt sich theoretisch eine bezüglich der Anforderungserfüllung optimale Anwendungsarchitektur planen. In der Realität gibt es teils erhebliche Einschränkungen hinsichtlich der Interoperabilität der Werkzeuge und des dafür nötigen Anpassungsaufwands. Dadurch kann es zu erheblichen Wechselwirkungen bei der Architekturplanung und ihrer Bewertung kommen. Die EAM-Ansätze berücksichtigen typischerweise jedoch nicht die Auswahl und Evaluation der Werkzeuge. Darüber hinaus stellt das Engineering hier einen Sonderfall dar, da selten integrative, monolithische Architekturen (vgl. Kap. 3.4.5) anzutreffen sind, sondern eine Vielzahl zu integrierender Werkzeuge. Bei ERP-Systemen sind idR. integrierte Architekturen anzutreffen, die auf einer gemeinsamen Datenbasis arbeiten und Standardschnittstellen aufweisen [FGS08, S. 1, 19]. Daher besteht in der Engineering-IT besonderer Bedarf zur Spezifikation der notwendigen Schnittstellen.

- **Fehlende ganzheitliche Sicht auf das Engineering:** Für das Management von IT-Architekturen und Organisationen werden Modelle eingesetzt (vgl. Kap. 3.4.8). Diese Modelle müssen jedoch alle relevanten Aspekte integrieren, um sie ganzheitlich abbilden zu können. Oftmals werden jedoch einzelne Aspekte nur unzureichend abgebildet und nicht miteinander in Beziehung gesetzt. Erschwerend kommt hinzu, dass eine Vielzahl von Stakeholdern mit verschiedenen fachlichen Hintergründen am Prozess der Entwicklung von Engineering-IT-Architekturen beteiligt ist. Von dem Modell der Engineering-IT-Architektur wird daher erwartet, dass es die Kommunikation zwischen den verschiedenen Rollen sicherstellen und als Spezifikation für die Anforderungen und die geplante Architektur dienen kann. Anwender erwarten klare Vorgaben, wie sie das Modell erstellen können.

Die Gegenüberstellung der Nutzenpotentiale und der Herausforderungen zeigt, dass die ganzheitliche, modellbasierte Gestaltung der Engineering-IT ein hohes Potential für die Befähigung des Engineerings darstellt. Eine so ausgewählte IT-Architektur erfüllt die funktionalen und qualitativen Anforderungen bestmöglich und erhöht die Datenverfügbarkeit und Durchgängigkeit. Es fehlt allerdings bisher an systematischer Unterstützung zur Erarbeitung und Modellierung der Modelle. Etablierte Ansätze wie das EAM berücksichtigen die besonderen Charakteristika des Engineerings nur unzureichend und haben Lücken hinsichtlich der abzubildenden Modellelemente. Darüber hinaus müssen die Ansätze oft erst auf den jeweiligen Anwendungsfall aufwändig angepasst werden. Aus diesen Gründen besteht ein **Bedarf** für eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*. Die Spezifikationstechnik soll dabei folgende Bestandteile enthalten.

- **Methode zur Analyse der Engineering-Prozessebene:** Die Spezifikationstechnik muss die systematische und methodisch unterstützte Analyse des Engineerings zur Modellierung von Prozessen, Methoden und der Artefakte ermöglichen. Dabei müssen sowohl Bewertungskriterien für die funktionalen Anforderungen aus fachlicher Sicht als auch die Anforderungen an die Gebrauchstauglichkeit aus Nutzersicht abgeleitet werden. Die Methode integriert sich in das Modellierungsrahmenwerk, um die Erfüllung der Grundsätze ordnungsmäßiger Modellierung zu unterstützen.
- **Methode zur Planung und Evaluierung von Engineering-IT-Architekturen:** Basierend auf den Architekturen der Prozessebene muss die Spezifikationstechnik die Ableitung geeigneter Anwendungsarchitekturen anleiten. Die Methode soll insbesondere auch die Spezifikation der resultierenden Schnittstellen zwischen den IT-Werkzeugen unterstützen. Zudem muss die Evaluierung der abgeleiteten Anwendungsarchitekturen anhand der auf Prozessebene definierten Bewertungskriterien ermöglicht werden.
- **Modellierungsrahmenwerk:** Das Modellierungsrahmenwerk ist eng verzahnt mit den genutzten Methoden und stellt eine geeignete Sprache zur Abbildung der entstehenden Artefakte bereit. Die Sprache soll dabei als domänenübergreifendes Kommu-

nikationsmedium sowie zur semi-formalen Spezifikation der Architekturen geeignet sein. Zur Vereinfachung der Modellierung werden entsprechende Sichten und für die Modellierung geeignete Werkzeuge benötigt.

- **Vorgehensmodell:** Es bedarf eines Vorgehensmodells, welches ein strukturiertes Vorgehen zur Entwicklung bewerteter Engineering-IT-Architekturen beschreibt. Dazu integriert es die notwendigen Tätigkeiten und Hilfsmittel zu einem konsistenten Handlungsstrang.

3.6 Anforderungen

Aus der Problemanalyse ergeben sich folgende Anforderungen an eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*. Die Anforderungen A1 bis A5 stellen spezifische Anforderungen an eine Spezifkationstechnik für Engineering-IT-Architekturen dar. Die Anforderungen A6 bis A8 sind allgemeine Anforderungen an Spezifikationstechniken.

A1) Fokus auf das Engineering: Die Spezifikationstechnik unterstützt die Anwendung im Engineering. Für diesen Anwendungsfall sind keine Anpassungen ohne Anleitung für die Anwender notwendig.

A2) Analyse der Prozess-, Methoden- und Datenarchitektur: Die Identifizierung, Analyse und Dokumentation der Engineering-Prozesse, Methoden und der resultierenden Artefakte werden systematisch und methodisch unterstützt. Dies umfasst sowohl die Identifikation der jeweiligen Objekte als auch deren Beziehungen.

A3) Identifikation und Analyse von Anforderungen: Die Methode soll die Identifikation und Analyse der funktionalen und qualitativen Anforderungen an die IT-Architektur unterstützen. Dabei sind geeignete Bewertungskriterien für eine IT-Architektur festzulegen. Dabei muss sowohl die fachliche Perspektive als auch die des Nutzers abgedeckt sein.

A4) Ableiten geeigneter Applikationsarchitekturen: Die Spezifikation zur Unterstützung der Prozess-Ebene geeigneter Applikationsarchitekturen soll methodisch angeleitet werden. Dabei müssen implementierbare Softwarekomponenten definiert und die notwendigen Schnittstellen spezifiziert werden.

A5) Anleitung zur Architektur-Evaluierung: Die Evaluierung spezifizierter Applikationsarchitekturen wird durch die Spezifikationstechnik unterstützt. Dabei sollen sowohl Rückschlüsse auf die Evaluierungsergebnisse für die gesamte Applikationsarchitektur wie auch einzelner Softwarekomponenten möglich sein. Zur Evaluierung müssen geeignete Kriterien bereitgestellt werden.

A6) Modellierungsrahmenwerk zur Spezifikation: Die Spezifikationstechnik stellt eine Sprache mit beschriebener Syntax und Semantik bereit, die sämtliche relevanten Aspekte innerhalb eines integrierten Modells abdecken kann. Die Sprache muss für alle in der Engineering-IT vertretenen Rollen verständlich sein und den Grundsätzen ordnungsmäßi-

ger Modellierung genügen. Zur Erstellung des Modells sollen verschiedene standardisierte Sichten bereitgestellt werden.

A7) Werkzeugunterstützung: Zur Erstellung und Verwaltung des Modells soll eine entsprechende Werkzeugunterstützung bereitgestellt werden. Diese soll das Metamodell der definierten Sprache beinhalten und, soweit technisch möglich und sinnvoll, die definierte Syntax und Semantik berücksichtigen. Darüber hinaus stellt die Werkzeugunterstützung notwendige Diagrammtypen und Konzepte zur Modellverwaltung bereit.

A8) Vorgehensmodell: Der Anwender wird durch ein systematisches und methodisches Vorgehensmodell in der Nutzung der Spezifikationstechnik angeleitet. Dabei wird insbesondere eine sachlogische Reihenfolge vorgestellt.

4 Stand der Technik

Das folgende Kapitel diskutiert Ansätze aus dem Stand der Technik und analysiert diese vor dem Hintergrund der Anforderungen aus Kapitel 3.6. In Kapitel 4.1 werden zunächst Ansätze für die domänenübergreifende, ganzheitliche Modellierung von Architekturen vorgestellt. Kapitel 4.2 stellt Ansätze zur Analyse von aus der Prozessebene resultierenden Anforderungen vor. Methoden zur Definition geeigneter Architekturen für die Engineering-IT werden in Kapitel 4.3 thematisiert. Kapitel 4.4 stellt Methoden zur Auswahl von IT-Werkzeugen als Bestandteile der IT-Architektur vor. Der somit dargelegte Stand der Technik wird in Kapitel 4.5 mit den Anforderungen abgeglichen. Daraus ergibt sich der Handlungsbedarf für die vorliegende Arbeit.

4.1 Ganzheitliche Ansätze zur Architekturbeschreibung

Nachfolgend werden verschiedene Ansätze zur Modellierung von (Teil-)Aspekten der Prozess-, Methoden- und/oder Datenarchitektur vorgestellt. Dabei werden neben einem allgemeinen Ansatz zur Definition von Architekturbeschreibungen konkrete Sprachen zur Modellierung von Prozessen, Methoden und Daten- bzw. Prozessarchitekuren vorgestellt.

4.1.1 Beschreibung von Architekturen nach ISO/IEC/IEEE 42010

Die ISO/IEC/IEEE 42010 – Systems and Software Engineering – Architecture Description beschreibt grundlegende Begriffe und ihre Abhängigkeiten, um Architekturbeschreibungen organisieren und ausdrücken zu können. Der Standard ist somit eine wichtige Grundlage für viele Modellierungsansätze von Architekturen. Er differenziert dabei zwischen Architekturen und der Architekturbeschreibung. Eine Architektur ist eine Eigenschaft eines Systems und definiert als [ISO42010, S. 2]:

> *„Fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution.“*

Eine Architekturbeschreibung ist ein Arbeitsergebnis (Artefakt), welches eine Architektur beschreibt. Kern des Standards ist die in Bild 4-1 dargestellte Ontologie.

Durch die Ontologie definiert der Standard grundlegende Konzepte zur Erstellung von Architekturbeschreibungen. Es gibt verschiedene *Stakeholder* eines Systems, die ein oder mehrere *Concerns*, ein spezielles Interesse oder Anliegen hinsichtlich des Systems, aufweisen. Die Concerns können auch mehreren Stakeholdern zugeordnet sein und entlang des Systemlebenszyklus verortet werden. Beispiele können der Informationsbedarf hinsichtlich Funktionalität, Struktur, Sicherheit oder Kosten des Systems darstellen. Eine Architekturbeschreibung enthält eine oder mehrere Sichten. Eine *Sicht* (View) beschreibt die Architektur aus der Perspektive eines Concerns auf ein spezifisches System. Nach

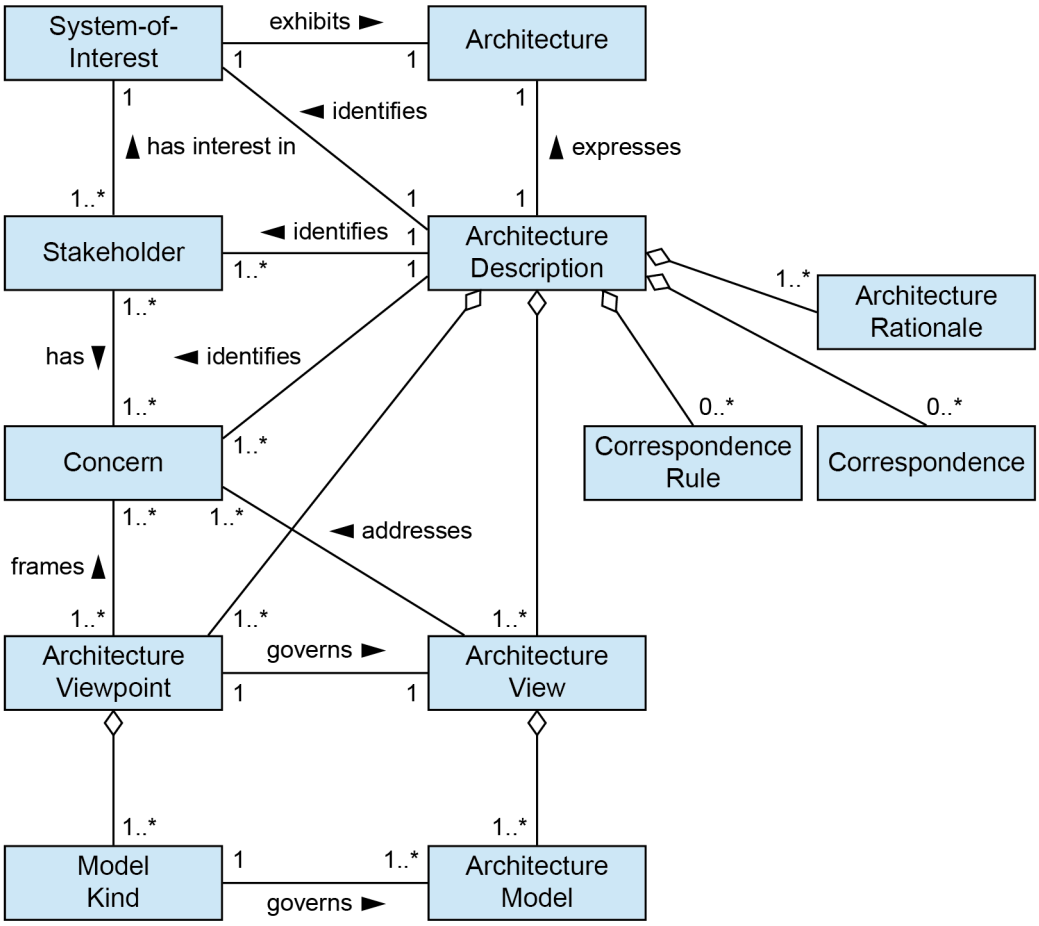


Bild 4-1: Ontologie zur Konzeptionalisierung von Architekturbeschreibungen nach [ISO42010, S. 5]

der ISO/IEC/IEEE 42010 beschreibt der Begriff der Architektur ein ganzheitliches, zusammenhängendes Konstrukt, worauf z. B. im Kontext der Enterprise Architektur eine Geschäfts-, Applikations- oder Datensicht eingenommen werden kann. Erst eine *Sichtweise* (Viewpoint) definiert die allgemeinen Konventionen für die Erstellung, Interpretation und Nutzung von Sichten, um bestimmte Concerns zu adressieren. Diese Konventionen können bspw. Sprachen, Notationen, Modellierungsmethoden oder Modellarten umfassen. Eine Sicht kann ein oder mehrere Architekturmodelle beinhalten. *Architekturmodelle* unterliegen auch hinsichtlich der *Modellart* (Model kind) bestimmten Modellierungskonventionen. Beispiele für Modellarten sind Datenfluss- oder Klassendiagramme [ISO42010, S. 3ff].

Architekturbegründungen (Architecture rationale) beschreiben, warum eine bestimmte Entscheidung innerhalb einer Architektur getroffen wurde, z. B. bei einer Entscheidung für eine von mehreren Alternativen. Eine *Correspondence* beschreibt Beziehungen zwischen Elementen einer Architekturbeschreibung, wobei diese durch *Correspondence Rules* zu definierenden Regeln unterliegen können [ISO42010, S. 7].

Der Standard nutzt drei wesentliche Konstrukte für Architekturbeschreibungen. Neben den Sichtweisen werden dazu Architektur-Rahmenwerke (Architecture frameworks) und Architekturbeschreibungs-Sprachen definiert. Zur Umsetzung dieser drei Kernkonstrukte

sind jeweils Anforderungen festgelegt. Ein **Architektur-Rahmenwerk** beschreibt Regeln und Vorgehensweisen für die Beschreibung von Architekturen in einem spezifischen Anwendungsbereich. So muss ein Rahmenwerk bspw. ein oder mehrere Stakeholder und Concerns sowie zugehörige Sichtweisen umfassen [ISO42010, S. 1f, 16ff]. Ein Beispiel für ein Architektur-Rahmenwerk stellt TOGAF dar [Mat11, S. 158], [BNM14, S. 3301]. Als **Architekturbeschreibungs-Sprache** (Architecture Description Language (ADL)) wird eine Ausdrucksform zur Architekturbeschreibung definiert. Dazu werden eine oder mehrere Modellarten bereitgestellt. Ein Beispiel für eine ADL stellt die OMG SysML dar [ISO42010, S. 10].

Bewertung: Der Standard definiert in einem geschlossenen Rahmen geeignete Konstrukte zur Beschreibung von Architekturen. Der Architektur- und Systembegriff ist dabei allgemeingültig und somit auch auf die vorliegende Problemstellung im Rahmen dieser Arbeit anwendbar. Es werden Anforderungen an die definierten Konstrukte gestellt, wodurch der Standard die Erarbeitung geeigneter Architekturbeschreibungen unterstützt und vereinheitlicht. Dadurch kann der Standard bei der Ableitung von Anforderungen an ein Architektur-Framework für die Gestaltung der Engineering-IT unterstützen. Es werden jedoch ausdrücklich keine Vorgehensweisen, Methoden, Notationen oder Techniken für die Erstellung von Architekturbeschreibungen vorgeben [ISO42010, S. 5]. Diese müssen separat bereitgestellt werden. Eine unmittelbare Anwendung des Standards für die Modellierung in einem spezifischen Anwendungsbereich ist nicht möglich und vorgesehen.

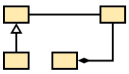
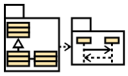
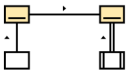
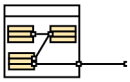
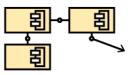
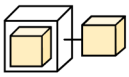
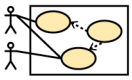
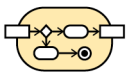
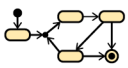
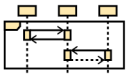
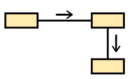
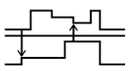
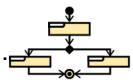
4.1.2 Sprachen zur Beschreibung von Architekturen

Unified (UML) und Systems (SysML) Modeling Language

Die **Unified Modeling Language (UML)** ist eine graphische Modellierungssprache zur Beschreibung von Softwaresystemen. Sie ist aktuell in der Version 2.5.1 verfügbar [OMG17-ol] und in der Version 2.4.1 in der ISO/IEC 19505-1 und ISO/IEC 19505-2 als internationaler Standard erschienen [ISO19505-1], [ISO19505-2]. Die UML umfasst eine Semantik und Syntax, die in einem Metamodell beschrieben sind. Sie definiert drei abstrakte Grundkonstrukte: *Classifier* klassifizieren Objekte hinsichtlich ihrer Merkmale, *Ereignisse* beschreiben hinsichtlich Ort bzw. Zeit messbare Erscheinungen und *Verhalten* beschreibt ausführbare Aktionen (z. B. als Reaktion auf ein Ereignis). Basierend auf diesen Grundtypen können spezifische Sprachkonstrukte mit einer detaillierteren Semantik in dreizehn verschiedenen Diagrammen genutzt werden. Diese sind in Tabelle 4-1 zusammengefasst [OMG17-ol, S. 1, 12f, 685], [OB09, S. 384ff].

Da die Sprachelemente eher allgemein gehalten sind, bietet die UML einen sogenannten **Profilmechanismus** an. Durch *Stereotypen* können bestehende Sprachelemente der UML um zusätzliche Eigenschaften, Semantik und konkrete Syntax erweitert werden. Stereotypen werden durch «Guillemets» kenntlich gemacht und in einem Profildiagramm definiert. Eine Menge von Stereotypen kann in einem *Profil* zusammengefasst werden [OMG17-ol, S. 278ff], [Wei14, S. 300f].

Tabelle 4-1: Übersicht über die Diagramme der UML nach [OMG17-ol, S. 685], [OB09, S. 243], [RQ12, S. 11f, 105ff]

	Diagramm	Beschreibung	Primäre Elemente
Strukturdiagramme	Klassendiag. 	Beschreibt die Struktur, Zusammenhänge und Attribute der Daten („Klassen“)	Klasse, Enumeration, Port, Assoziation
	Paketdiag. 	Strukturiert das Modell in hierarchische bzw. modulare Bestandteile	Paket, Import- bzw. Mergebeziehung
	Objektdiag. 	Beschreibt die Instanzen eines Systems zu einem bestimmten Zeitpunkt	*Greift auf andere Diagrammtypen zurück*
	Kompositionsstrukturdiag. 	Beschreibt die interne Struktur von Objekten (Klassen) und ihre Beziehungen	Part, Port, Collaboration Use, Konnektor
	Komponentendiag. 	Beschreibt die Struktur des Systems zur Laufzeit	Komponente, Beziehung, Schnittstelle
	Verteilungsdiag. 	Beschreibt die Zuordnung/Verteilung von Software- auf Hardwareeinheiten	Knoten, Artefakt, Einsatzspezifikation
Verhaltensdiagramme	Use Case Diag. 	Beschreibt das externe Systemverhalten aus Nutzersicht	Anwendungsfall, Aktor, Assoziationen
	Aktivitätsdiag. 	Beschreibt detailliert (verzweigte) Abläufe des Systemverhaltens	Aktivität, Knoten, Swimlanes
	Zustandsdiag. 	Beschreibt Zustände und dazugehörige Übergangsbedingungen eines Objektes	Zustand, Transition, Verzweigungen
	Sequenzdiag. 	Beschreibt das dynamische Zusammenspiel zwischen Objekten	Nachricht, Lebenslinie, Objekte
	Kommunikationsdiag. 	Beschreibt das dynamische Zusammenspiel in komplexen Strukturen	Nachricht, Lebenslinie
	Timingdiag. 	Beschreibt das zeitliche Verhalten von Objekten	Nachricht, Lebenslinie, (Zeit-)bedingung
	Interaktionsübersichtsdiag. 	Beschreibt durch Abfolgen das Zusammenspiel mehrere Interaktionen	Aktivität, Knoten, Swimlanes

Die **Systems Modeling Language (SysML)** ist eine graphische Modellierungssprache zur Modellierung technischer Systeme im Rahmen des Systems Engineerings. Insbesondere sollen die interdisziplinäre Spezifikation von Anforderungen, Strukturen, Verhalten, Allokationen und Randbedingungen für Systemeigenschaften unterstützt werden. Sie ist in der Version 1.6 durch die OMG spezifiziert [OMG19-ol] und in der Version 1.4 als ISO/IEC 19514 standardisiert [ISO19514]. Die SysML nutzt Teile der UML und erweitert diese durch ein Profil. Zur Modellierung werden die in Bild 4-2 beschriebenen neun Diagrammtypen bereitgestellt [OMG19-ol, S. 1, 7, 211].

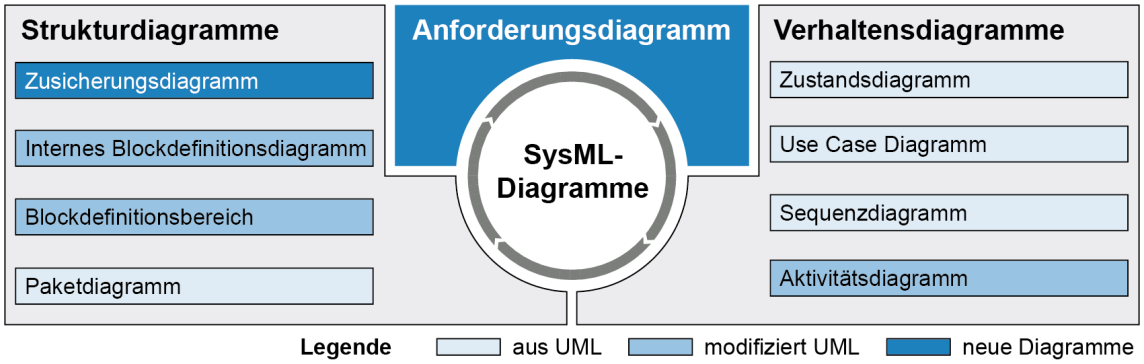


Bild 4-2: Diagramme der SysML in Anlehnung an [DGO+14, S. 37], [OMG19-ol, S. 211f]

- Die **Strukturdiagramme** beschreiben statische, strukturelle Eigenschaften des Systems [Alt12, S. 40]. Das *Blockdefinitionsdiagramm* definiert Blöcke als Repräsentationen von Systemelementen und strukturiert diese hierarchisch mit Hilfe von Assoziationen. Die Beziehungen zwischen Blöcken werden in einem *internen Blockdiagramm* durch Ports, Flüsse und Konnektoren dargestellt. Das *Zusicherungsdiagramm* erlaubt die Definition parametrischer Beziehungen zwischen Eigenschaften von Systemelementen mit Hilfe von Constraint Blöcken. *Paketdiagramme* erlauben die Strukturierung des Modells [FMS12, S. 30], [Wei14, S. 336ff].
- Das **Anforderungsdiagramm** erlaubt die Strukturierung textueller Anforderungen. Dazu werden verschiedene Beziehungstypen (z. B. Erfüllt durch, Verifiziert durch, Verfeinert durch) bereitgestellt. Die Beziehungen können je nach Typ zwischen Anforderungen oder zu anderen Modellelementen erstellt werden. Typische Darstellungsformen sind Grafiken oder Tabellen [OMG19-ol, S. 186ff], [FMS12, S. 30].
- Durch die vier **Verhaltensdiagramme** können dynamische Eigenschaften des Systems, das Verhalten, beschrieben werden [Alt12, S. 40]. *Use Case Diagramme* beschreiben die Nutzung von Systemfunktionen durch externe Akteure. *Aktivitätsdiagramme* definieren Abläufe von Aktivitäten, die ggf. verzweigt oder parallel verlaufen und die dazu benötigten In- und Outputs. *Sequenzdiagramme* beschreiben den Austausch von Nachrichten (Kommunikationsabläufe) zwischen Systemen bzw. ihrer Elemente. *Zustandsdiagramme* beschreiben Zustände eines Systems sowie deren Übergänge durch Transitionen mit den auslösenden Bedingungen bzw. Ereignissen [Wei14, S. 359ff], [FMS12, S. 30], [Alt12, S. 49ff].

Bewertung: Die SysML ist eine weit verbreitete Sprache zur Modellierung interdisziplinärer Systeme, während die UML den Fokus auf Softwaresysteme legt. Neben einer definierten Syntax und Semantik sind verschiedene Diagrammtypen definiert, die es erlauben Anforderungen, Anwendungsfälle und Architekturen von Systemen zu modellieren. Sowohl UML als auch SysML sind reine Sprachen, sie stellen daher keine Methode bereit. Beide Sprachen sind relativ abstrakt gehalten und decken nicht die notwendigen semanti-

schen Differenzierungen zur Abbildung einer Engineering-IT-Architektur ab. Die konkrete Syntax bietet der Spezifikation nach nur geringe Differenzierungsmerkmale hinsichtlich der Semantik. Da UML die technische Grundlage darstellt, ist jedoch eine Erweiterung der Semantik über den Profilmechanismus möglich. Zudem werden beide Sprachen durch eine Vielzahl von Modellierungswerkzeugen unterstützt.

Spezifikationstechnik CONSENS

Die am Heinz Nixdorf Insitut entwickelte Spezifikationstechnik CONSENS (CONceptual design Specification technique for the Engineering of complex Systems) ist ein Ansatz zur ganzheitlichen und disziplinübergreifenden Beschreibung technischer Systeme. Der Schwerpunkt liegt auf der Unterstützung der Konzeptphase. CONSENS umfasst eine Modellierungsmethode und eine darauf abgestimmte Sprache, die in sieben Partialmodelle gegliedert wird (s. Bild 4-3). Die einzelnen Aspekte werden im Wechselspiel erstellt, es ist jedoch eine gewissen Abfolge vorgesehen [GFD+08, S. 91f]. Das Vorgehen gliedert sich in eine Phase zur *Analyse* des Systems und daraus resultierenden Anforderungen. In der darauffolgenden Phase zur *Synthese* wird eine geeignete Lösung aus einzelnen Elementen zusammengesetzt. Sie spiegelt damit eine systematische, methodische Arbeitsweise nach FELDHUSEN UND GROTE wieder [FG13, S. 285].

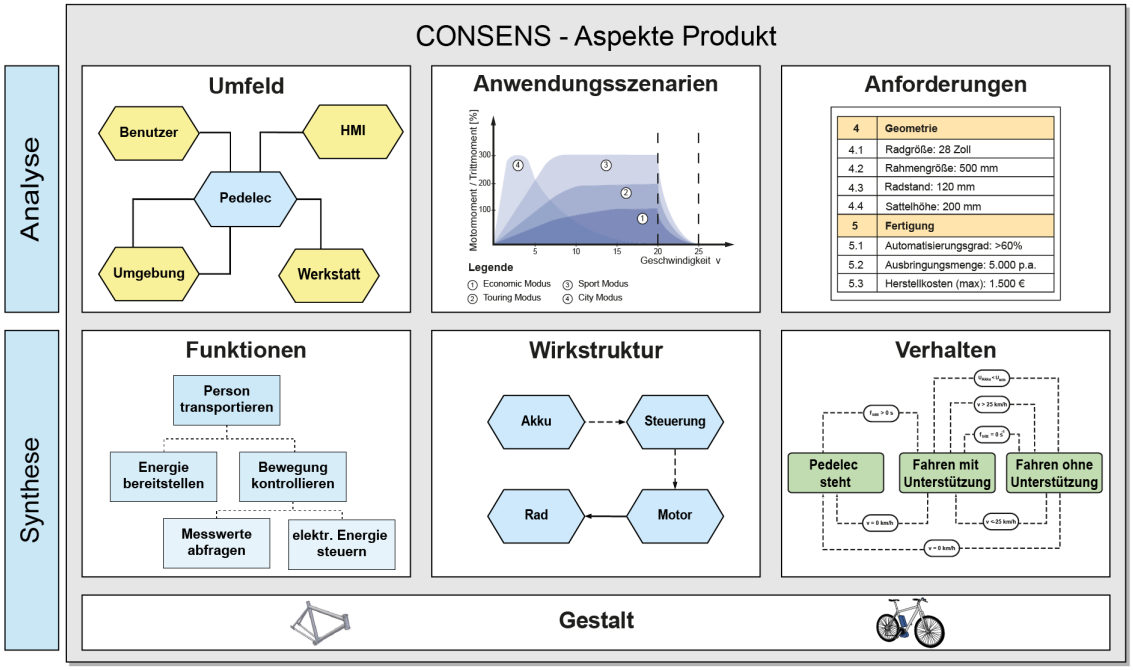


Bild 4-3: Partialmodelle nach CONSENS zur Spezifikation mechatronischer Systeme nach [GDE+18, S. 419]

In der **Analysephase** wird zunächst mit Hilfe des *Umfeldmodells* die Systemgrenze festgelegt. Darüber hinaus werden Schnittstellen zwischen dem System und dessen Umwelt identifiziert. Zudem wird mit Hilfe der *Anwendungsszenarien* das gewünschte Verhalten des Systems aus Nutzersicht beschrieben. Sie werden entlang des PLZ aufgenommen und

betrachten unterschiedliche Situationen. Aus beiden Partialmodellen werden anschließend *Anforderungen* abgeleitet [GDE+18, S. 412, 419ff], [DDG+14, S. 119ff].

In der **Synthesephase** werden die Anforderungen analysiert und notwendige *Systemfunktionen* als Substantiv-Verb-Kombinationen abgeleitet. Diese werden in einer Hierarchie strukturiert. In einem folgenden Schritt werden den Funktionen mögliche Lösungsmuster zugeordnet, was durch Methoden wie den morphologischen Kasten nach ZWICKY [FG13, S. 372ff] oder die Konsistenzanalyse nach KÖCKERLING [Köc04, S. 79ff] unterstützt werden kann [GDE+18, S. 422]. Die Lösungsmuster werden anschließend in Systemelemente überführt und in der *Wirkstruktur* synthetisiert. Diese beschreibt die Systemelemente und ihre Wirkzusammenhänge in Form von Energie-, Stoff- und Informationsflüssen sowie mechanischen Verbindungen und logischen Beziehungen. Davon ausgehend wird das *Verhalten* durch Zustands- und Aktivitätsdiagramme beschrieben, die an die SysML angelehnt sind (vgl. Kap. 4.1.2). Die *Gestalt* wird für gestaltbehaftete Systemelemente im Rahmen erster geometrischer Festlegungen beschrieben und typischerweise in Form von CAD-Modellen definiert [GDE+18, S. 412, 421ff], [DDG+14, S. 123ff].

Bewertung: Die Spezifikationstechnik CONSENS nutzt Grundprinzipien des ingenieurmäßigen Vorgehens und des SE zur Entwicklung von technischen Systemen und berücksichtigt dabei sowohl die systematische Identifikation und Analyse von Anforderungen als auch die Definition geeigneter Architekturen. Sie stellt dazu ein Vorgehen mit methodischer Unterstützung bereit. Darüber hinaus legt die Methode fest, welche der bereitgestellten, intuitiv nutzbaren Sprachelemente zur Darstellung spezifischer Sachverhalte genutzt werden. CONSENS fokussiert jedoch die Entwicklung technischer Systeme und berücksichtigt somit nicht die Spezifika der Engineering-IT. Die Evaluierung von Architekturen wird nicht betrachtet.

ArchiMate Modellierungssprache für Enterprise Architekturen

Die Modellierungssprache ArchiMate ist eine von der Open Group spezifizierte graphische Modellierungssprache. Sie zielt im Gegensatz zu den zuvor beschriebenen Ansätzen auf die Modellierung von Enterprise Architekturen ab. Zur Lesbarkeit durch verschiedene Stakeholder wird eine eingängige Notation sowie die nötige Semantik und Syntax bereitgestellt. ArchiMate orientiert sich an der Architekturdefinition und den Konzepten von TOGAF (s. Kap. 4.3.1) [TOG19, S. 173f], [Mat11, S. 68]. ArchiMate beschreibt, wie in Bild 4-4 dargestellt, sechs **Ebenen**, die den Schritten der ADM von TOGAF zugeordnet werden können. Darüber hinaus werden vier **Aspekte** betrachtet: Die *aktive Struktur* (Struktur der Elemente, die Verhalten aufweisen bzw. ausführen, z. B. Anwendungskomponenten), das *Verhalten* (beschreibt das Verhalten, z. B. als Prozess) und die *passive Struktur* (Struktur der Objekte, die durch das Verhalten bearbeitet werden, z. B. Daten- oder physische Objekte). Elemente zur Dokumentation einer *Motivation* oder Begründung mit Einfluss auf das Architekturmodell stellen einen eigenen Aspekt dar, z. B. Stakeholder, Anforderungen oder Randbedingungen [TOG19, S. 8ff, 40ff]. Darüber hinaus steht ein Mechanismus zur

Anpassung der Sprache zur Verfügung, um Attribute zu ergänzen oder neue Elemente als Spezialisierung existierender Sprachelemente anzulegen [TOG19, S. 116ff], [Lan17, S. 110f].

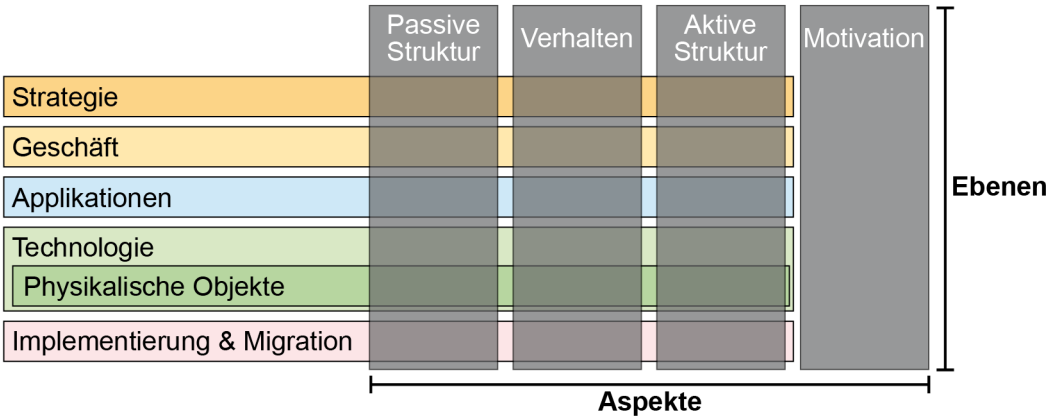


Bild 4-4: Das ArchiMate Framework nach [TOG19, S. 9]

- Elemente zur Dokumentation einer Motivation oder Begründung mit Einfluss auf das Architurmodell werden mit dem **Motivationsaspekt** über alle Ebenen hinweg festgehalten. Wesentliche Objekte sind *Stakeholder, Treiber* (z. B. Concerns eines Stakeholders) und *Bewertungen*. Zielzustände können über *Ziele, Ergebnisse, Prinzipien, Anforderungen* und *Einschränkungen* definiert werden. *Bedeutungen* (Interpretation eines Modellelements) und *Werte* können andere Modellelemente näher spezifizieren.
- In der **Strategieebene** werden architekturrelevante strategische Entscheidungen und Ausrichtungen modelliert, z. B. durch *Ressourcen, Fähigkeiten* oder *Wertströme* [TOG19, S. 51ff].
- Die **Geschäftsebene** umfasst Sprachkonstrukte zur Modellierung der operationellen Organisation. Für die Beschreibung der aktiven Struktur sind dies u. a. *Akteure, Rollen, Kollaborationen* und *Schnittstellen*. Das Verhalten kann durch *Geschäftsprozesse, -funktionen, -interaktionen, -ereignisse und -dienstleistungen* beschrieben werden. Die passive Struktur wird durch *Geschäftsobjekte, Verträge* und *Repräsentationen* spezifiziert [TOG19, S. 58ff].
- Mit der **Applikationsebene** werden Applikationsarchitekturen beschrieben. *Applikationskomponenten, -kollaborationen, und -schnittstellen* beschreiben die aktiven Strukturelemente. Das Verhalten wird durch *Applikationsfunktionen, -interaktionen, -prozesse, -ereignisse* sowie *-services* charakterisiert. *Datenobjekte* definieren die passiven Strukturelemente [TOG19, S. 73ff].
- Die Technologiearchitektur einer Organisation wird in der **Technologieebene** konkretisiert. Aktive Strukturelemente können durch *Knoten, Geräte, Systemsoftware, Technologiekollaborationen* und *-schnittstellen, Pfade* und *Kommunikationsschnittstellen* beschrieben werden. Für das Verhalten stehen *Technologiefunktionen, -prozesse,*

-interaktionen, -ereignisse und *-dienstleistungen* zur Verfügung. *Artefakte* definieren die passive Struktur [TOG19, S. 83ff].

- Die **physikalische Ebene** erweitert die Technologieebene um physikalisch existierende Objekte. Als aktive Strukturelemente stehen dazu *Anlagen, Werke* und *Verteilnetzwerke* bereit, für die passive Struktur *Materialien* [TOG19, S. 97ff].
- Die **Implementierungs- und Migrationsebene** unterstützt die Umsetzung der Architekturen. Zur Spezifikation stehen *Arbeitspakete, Deliverables, Implementierungsereignisse, Plateaus* (stabile Architekturstände) und *Lücken* (Differenz zwischen zwei Plateaus) bereit [TOG19, S. 105ff].

Bewertung: ArchiMate stellt eine Sprache bereit, die konkrete Sprachkonstrukte zur Beschreibung von Enterprise Architekturen umfasst. Dadurch grenzt sie sich maßgeblich von Sprachen wie der OMG UML ab, die eher abstraktere Elemente vorgeben. ArchiMate berücksichtigt zwar Aspekte der Applikations- und Datenarchitekturen, die Modellierung von Prozessarchitekturen jedoch nur eingeschränkt. So werden keine Methoden berücksichtigt und keine Prozessaktivitäten wie bspw. in Prozessmodellierungssprachen. Die Analyse von Anforderungen an die Engineering-IT ist daher nur eingeschränkt möglich. Die Evaluierung von Architekturen wird nicht berücksichtigt. Die Integration verschiedener Modellierungssprachen ist komplexer als bei den von der OMG spezifizierten Sprachen und setzt entsprechende Transformationen voraus [Lan17, S. 281]. Die Werkzeugunterstützung durch kommerzielle Werkzeuge ist zwar gegeben, jedoch nicht so breit wie bspw. für die UML.

4.1.3 Sprachen zur Beschreibung von Prozessen und Methoden

Objektorientierte Methode zur Geschäftsprozessmodellierung und -analyse (OMEGA)

Die objektorientierte Methode zur Geschäftsprozessmodellierung und -analyse (OMEGA) entstand im Rahmen der Forschung am Heinz-Nixdorf-Institut und wurde gemeinsam mit der Unternehmensberatung Unity AG weiterentwickelt [GP14, S. 254], [Fah95, S. 62ff]. Ziel von OMEGA ist die vollständige Modellierung der Ablauforganisation in einem Modell mit einer anschaulichen und prägnanten Notation. Somit eignet sich OMEGA als Instrument zur Analyse und Planung von Geschäftsprozessen. Die relevanten Sprachkonstrukte sind in Bild 4-5 dargestellt und werden nachfolgend vorgestellt [GP14, S. 254ff].

- Ein **Geschäftsprozess** ist eine Abfolge von zusammenhängenden Tätigkeiten zur Erbringung oder Transformation eines Objektes. Er kann durch einen Input als definierten Anfang und Output als definiertes Ende abgegrenzt werden.
- Durch **Organisationseinheiten** wird ein Element der Aufbauorganisation repräsentiert. Dieses führt den Geschäftsprozess aus bzw. ist für diesen verantwortlich.

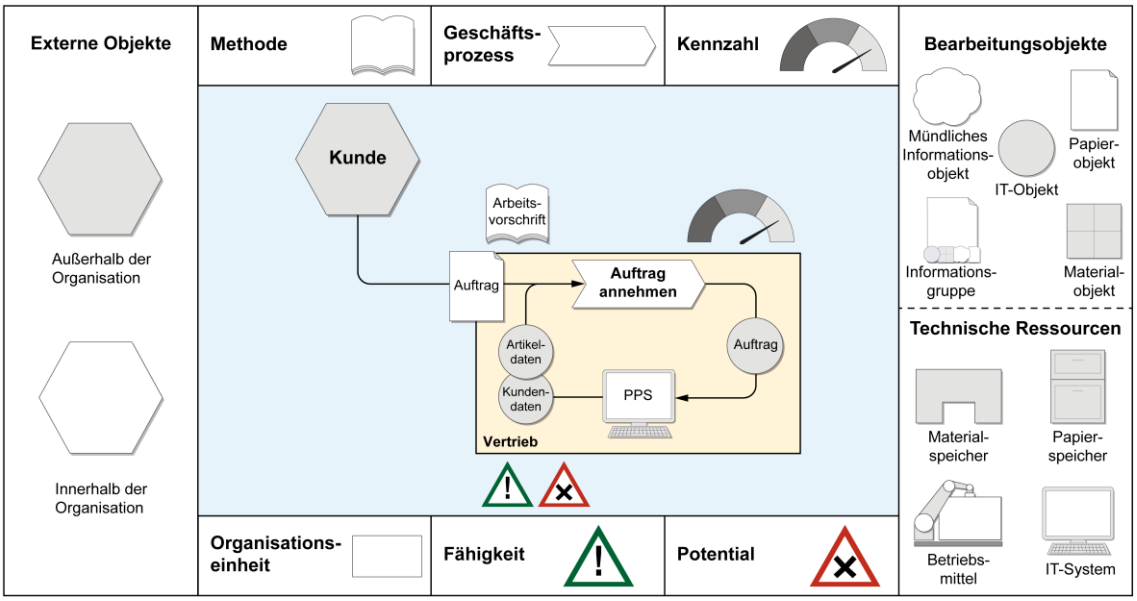


Bild 4-5: Überblick über die Konstrukte der Methode OMEGA [GP14, S. 254]

- **Externe Objekte** beschreiben Schnittstellen des Prozesses zu Einheiten außerhalb des Betrachtungsgegenstands. Es wird unterschieden zwischen externen Objekten, die *außerhalb der eigenen Organisation* und *innerhalb der eigenen Organisation, aber außerhalb des Untersuchungsbereichs* angeordnet sind.
- **Bearbeitungsobjekte** stellen die In- und Outputs von Geschäftsprozessen dar. OMEGA differenziert hier nach fünf Arten: *IT-Objekte* (z. B. E-Mails, CAD-Modelle), *Papierobjekte* (z. B. Formulare, gedruckte Zeichnungen), *mündliche Informationsobjekte* (z. B. Gespräche, Telefonate), *Materialobjekte* (z. B. Werkstück, Halbzeug) und *Informationsgruppen* als Menge verschiedener Bearbeitungsobjekte (z. B. Software-Paket inkl. Datenträger und Versanddokumente).
- Die Durchführung von Geschäftsprozessen kann bei Bedarf durch **technische Ressourcen** unterstützt werden. Dazu werden vier Arten unterschieden: *IT-Systeme* (z. B. CAD-Werkzeug, PDM-System); *Betriebsmittel* (z. B. NC-Fräse, Portalroboter); *Papierspeicher* (z. B. Archiv, Register) und *Materialspeicher* (z. B. Arbeitsplatzpuffer, Fertigteilelager).
- **Potentiale** kennzeichnen Verbesserungspotentiale in einem Prozess. **Fähigkeiten** weisen auf besondere vorhandene bzw. notwendige Fähigkeiten zur Durchführung eines Prozesses hin.
- Eine **Methode** beschreibt eine Abfolge von Arbeitsschritten zur Erzielung eines definierten Ergebnis. Sie unterstützt einen oder mehrere Geschäftsprozesse und ist diesen zugeordnet.

Geschäftsprozesse, externe Objekte und technische Ressourcen können durch **Kommunikationsbeziehungen** miteinander verbunden werden. Sie weisen immer einen Sender und

Empfänger auf. Wenn Bearbeitungsobjekte einer Kommunikationsbeziehung zugeordnet werden, drückt diese den Material- bzw. Informationsfluss aus. Durch **Entscheidungen** lassen sich logische Abhängigkeiten zwischen Prozessen darstellen. Für die Umsetzung existieren Modellierungsregeln und eine Werkzeugunterstützung durch ein proprietäres IT-Werkzeug und Workshopkarten.

Bewertung: OMEGA ist ein intuitiver Ansatz zur Modellierung von Geschäftsprozessen. Eine detaillierte Semantik, die Objekte wie Methoden oder technische Ressourcen umfasst, erlaubt eine tiefer gehende Analyse von Prozessen und daraus resultierender Anforderungen. Methoden können jedoch nicht im Detail beschrieben werden. Die Sprache umfasst ein Objekt für IT-Systeme, wobei jedoch existierende IT-Werkzeuge aufgenommen und keine Architektur abgebildet werden kann. Es existieren vielfach keine expliziten Beziehungen, diese werden lediglich räumlich ausgedrückt. Die Werkzeugunterstützung ist mit einem verfügbaren Modellierungswerkzeug stark eingeschränkt, eine Integration mit anderen Sprachen nicht ohne weiteres möglich.

Business Process Model and Notation (BPMN)

Die Business Process Model and Notation (BPMN) 2.0.2 ist eine durch die Object Management Group (OMG) entwickelte graphische Modellierungssprache zur Beschreibung von Geschäftsprozessen. Die Sprache umfasst Syntax und Semantik und ist in der ISO/IEC 19510 in der Version 2.0.1 standardisiert [ISO19510]. Sie ist mit dem Ziel geschaffen worden, einfach und intuitiv für alle Nutzer verständlich zu sein und die Lücke zwischen der Gestaltung von Prozessen und ihrer Implementierung zu füllen [OMG14-ol, S. 1]. Dazu werden die in Bild 4-6 dargestellten grundlegenden **Sprachkonstrukte** bereitgestellt. Diese sind nachfolgend in vier Kategorien beschrieben [OMG14-ol, S. 25ff].

- **Flussobjekte** beschreiben das Verhalten eines Geschäftsprozess und sind somit elementare Elemente in der Darstellung von Prozessen. *Aktivitäten* stellen Tätigkeiten oder Teilprozesse dar. *Ereignisse* repräsentieren, unterteilt in Start-, Zwischen- oder Endereignisse, ein relevantes Geschehen, welches Auslöser oder Resultat von Aktivitäten darstellen kann. Gateways können Prozesse verzweigen oder zusammenführen [OMG14-ol, S. 26f], [FR16, S. 32ff].
- **Daten** repräsentieren Informationen, unabhängig von ihrer physischen Beschaffenheit. *Datenobjekte* agieren als In- oder Outputs für Aktivitäten und können als Datensammlung auch eine Aggregation von Informationen umfassen. Datenin- und -outputs sind analog zu Datenobjekten zu betrachten, beziehen sich jedoch auf einen Prozess und nicht auf Aktivitäten. Datenobjekte können in *Datenspeichern* abgerufen oder abgelegt werden [OMG14-ol, S. 207], [All15, S. 134ff].
- **Verbindungsobjekte** erlauben die Verknüpfung von Modellelementen. *Sequenzflüsse* legen eine logische Reihenfolge fest, indem sie Flussobjekte innerhalb eines Pools verbinden. *Nachrichtenflüsse* tauschen Informationen poolübergreifend aus.

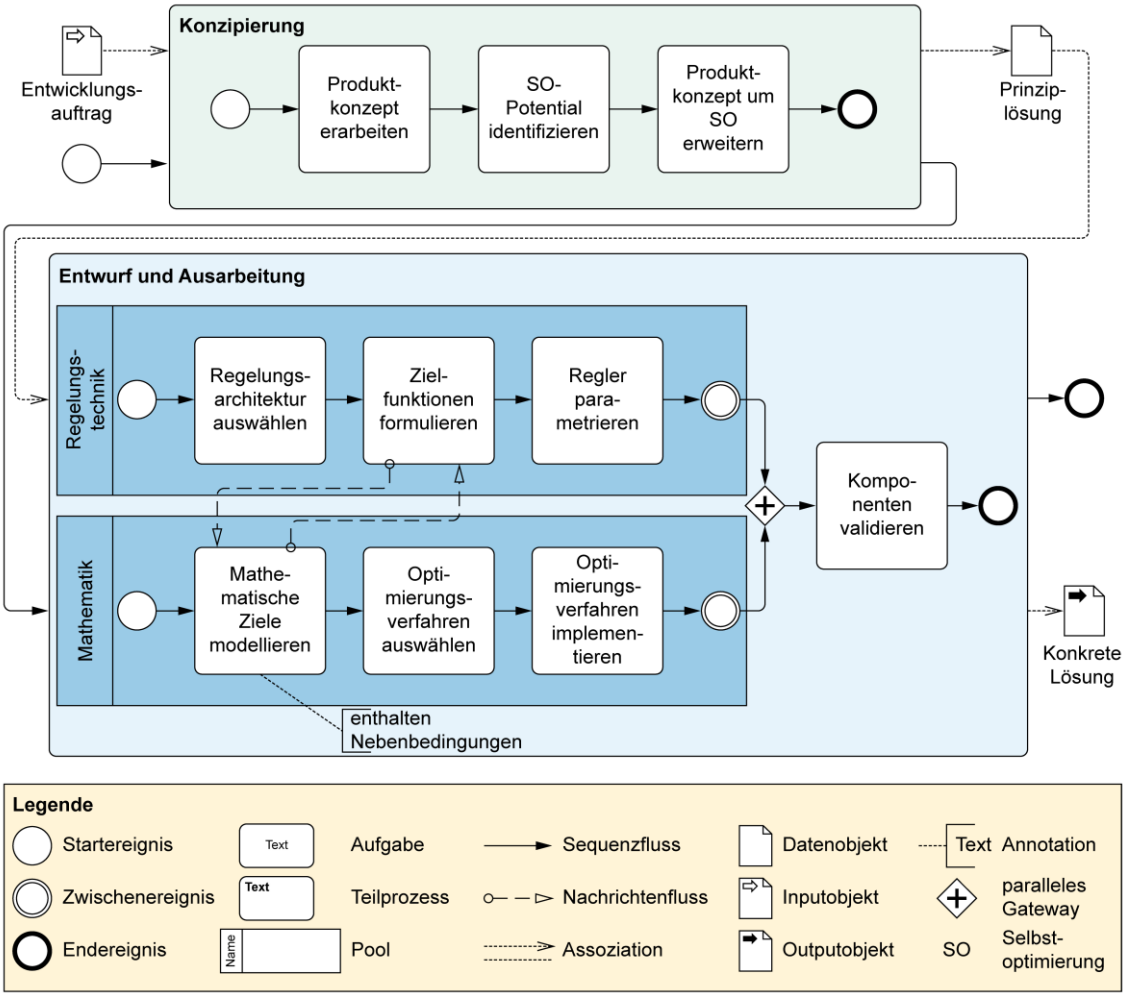


Bild 4-6: Beispiel eines BPMN-Prozessdiagramms und Übersicht der Sprachkonstrukte [GP14, S. 253]

Datenassoziationen verbinden Datenobjekte und Aktivitäten, wohingegen *Assoziationen* Artefakte mit anderen Modellelementen gerichtet oder ungerichtet verbinden [OMG14-ol, S. 27, 220], [FR16, S. 28f].

- **Swimlanes** ermöglichen die Gruppierung von Modellelementen über Verantwortungsbereiche. *Pools* repräsentieren eine Rolle (z. B. Organisationseinheit), die für die Steuerung des Prozesses verantwortlich ist. Der Pool stellt die Grenzen eines in sich abgeschlossenen Prozess dar und kann sowohl mit Prozessinhalten als auch als Black-Box modelliert werden. *Lanes* sind Sub-Strukturen eines Pools, um Aktivitäten zu organisieren oder zu kategorisieren [OMG14-ol, S. 27], [FR16, S. 92].
- **Artefakte** stellen zusätzliche Informationen zum Prozess bereit, beeinflussen den Prozessablauf jedoch nicht. BPMN stellt dazu *Gruppen* und *Textannotationen* bereit, sie können jedoch auch durch eigene Artefakttypen erweitert werden [OMG14-ol, S. 26ff], [FR16, S. 102f].

Zur Darstellung dieser Objekte stellt BPMN drei Arten von **Diagrammen** bereit. *Prozessdiagramme* beschreiben Sequenzen bzw. Flüsse von Aktivitäten innerhalb einer Organisation, bspw. wie in Bild 4.1.3 dargestellt [OMG14-ol, S. 1, 153]. *Choreographiediagramme* begrenzen die Betrachtung innerhalb eines Prozesses auf den Austausch von Nachrichten zwischen Rollen (sog. Teilnehmern) und zielen auf eine Formalisierung des Informationsaustauschs ab [OMG14-ol, S. 315], [All15, S. 11]. In *Kollaborationsdiagrammen* werden zwei oder mehr Pools als White- oder Black-Box dargestellt und der Informationsfluss zwischen diesen beschrieben. Konversationsdiagramme sind eine spezielle Unterart der Kollaborationsdiagramme, bei der die Pools als Black-Boxen betrachtet und die Nachrichten nicht durch detaillierte Choreographien, sondern mit vereinfachenden Konversationen beschrieben werden [OMG14-ol, S. 24, 110], [All15, S. 11f]. In der Praxis werden insbesondere Prozess-, aber auch Kollaborationsdiagramme am häufigsten eingesetzt [All15, S. 12], bei Kollaborationsdiagrammen wird stellenweise die Praxisrelevanz angezweifelt [FR16, S. 111].

Bewertung: BPMN bietet eine etablierte und weit verbreitete Modellierungssprache zur graphischen Modellierung von Geschäftsprozessen. Sie stellt hierzu eine benutzerfreundliche und intuitive Notation bereit. Dabei wird der Prozessablauf fokussiert, die Modellierung von Daten anhand der graphischen Beschreibungssprache ist in Grundzügen möglich. Eine Betrachtung der IT-Werkzeuge ist nicht vorgesehen. BPMN berücksichtigt keine Methoden oder physische Ressourcen. Eine Methode zur Modellierung ist nicht enthalten, da BPMN eine reine Modellierungssprache darstellt. Hervorzuheben ist die Standardisierung der Sprache, BPMN gilt als de-facto Standard und ist die von den meisten Modellierungswerkzeugen unterstützte Sprache [All15, S. 10] zur Prozessmodellierung.

Software & Systems Process Engineering Metamodel (SPEM)

Das Software & Systems Process Engineering Metamodel (SPEM) 2.0 ist ein von der OMG entwickeltes Metamodell zur Modellierung, Dokumentation und Implementierung von Engineering Prozessen und Methoden. Dazu werden ein Metamodell, ein Profil für die OMG UML2 mit definierter Syntax und Semantik (SPEM 2.0 Profil) sowie ein konzeptioneller Nutzungsrahmen bereitgestellt. SPEM gliedert sich in sieben thematisch aufgeteilte Pakete (z. B. Methodenbeschreibung, Methodenbibliotheken, etc.), welche SPEM modular aufbauen [OMG08-ol, S. 1f, 9f, 19]. Die Pakete und im Rahmen dieser Arbeit besonders relevante Sprachkonstrukte sollen im Folgenden kurz vorgestellt werden. Bild 4-7 und 4-8 zeigen die beispielhafte Anwendung von SPEM.

- **Kernelemente** (Core) definieren grundlegende, abstrakte Stereotypen, die in weiteren Paketen genutzt und verfeinert werden. Durch sogenannte *Typen* (Kind) können Instanzen bestimmter SPEM-Objekte benutzerdefiniert typisiert werden. Im SPEM 2.0 Base Plug-in sind dazu bereits verschiedene Typen vordefiniert [OMG08-ol, S. 35ff, 155ff].

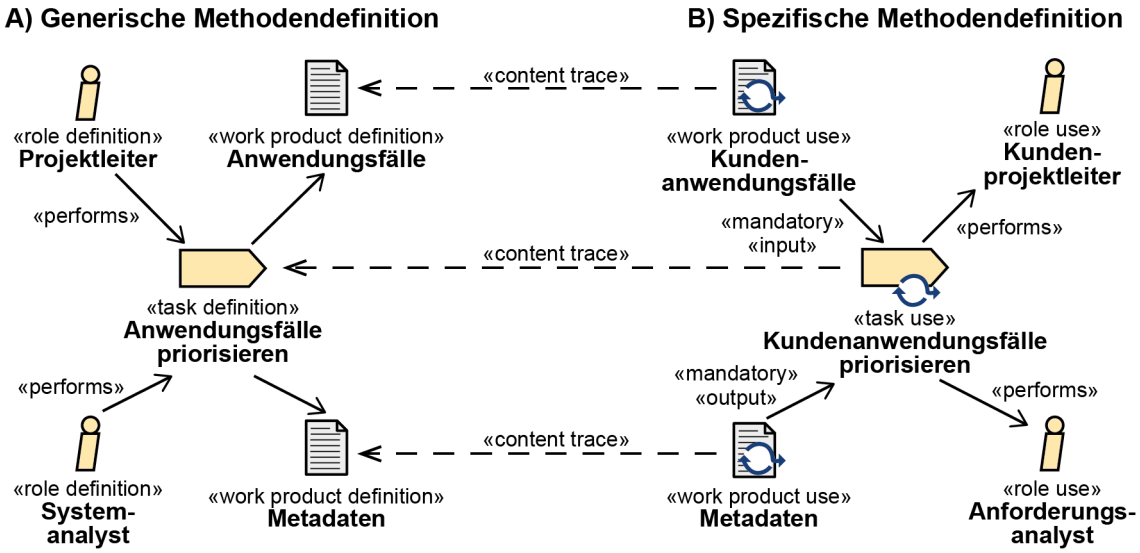


Bild 4-7: Beispiel einer mit SPEM definierten Methode und ihrer spezifischen Nutzung

- In dem Paket für **Methodenbeschreibungen** (Method Content) sind, wie in Bild 4-7-A dargestellt, die Kernelemente für die Definition von Methoden enthalten. *Aufgabendefinitionen* (Task Definition) beschreiben eine im Rahmen einer Methode durchzuführende Tätigkeit, die hinsichtlich ihrer Granularität einige Stunden bis Tage umfasst. Sie können durch *Schritte* (Step) verfeinert werden, die alternative Abläufe ermöglichen und nicht zwingend durchlaufen werden müssen. *Artefakte* (Work Product) stellen In- und Outputs von Aufgabendefinitionen dar. Sie können durch Aggregationen bzw. Kompositionen hierarchisiert werden, außerdem können Querbeziehungen zwischen Artefakten modelliert und als verpflichtender oder optionaler In- bzw. Output zu Aufgabendefinitionen zugeordnet werden. *Rollen* (Role) beschreiben Kompetenzen und Verantwortlichkeiten eines oder mehrerer Individuen. Über differenzierte Beziehungen können Rollen als verantwortlich oder ausführend einer Aufgabendefinitionen zugeordnet werden. *Werkzeuge* (Tool Definition) unterstützen die Durchführung von Aufgabendefinitionen und können mit diesen verknüpft werden [OMG08-ol, S. 81ff].

- Das Paket für **Prozessstrukturen** (Process Structure) erlaubt die Beschreibung von linearen und iterativen Entwicklungsprozessen und ihrer Strukturen. *Aktivitäten* (Activities) beschreiben Arbeitsschritte oder ganze Geschäftsprozesse. Sie können hierarchisch aufgebaut werden, zu sequentiellen Abfolgen verknüpft und durch einen Wiederverwendungsmechanismus im Kontext spezifischer Aktivitäten in angepasster Form wiederverwendet werden. Bestandteile einer Methode können im Rahmen von Prozessen im spezifischen Kontext einer Aktivität genutzt werden. *Spezifische Rollen* (Role Use) und *Spezifische Artefakte* (Work Product Use) repräsentieren die Nutzung einer Rolle bzw. eines Artefakts im spezifischen Kontext einer Aktivität. Sie sind über Beziehungen mit der Aktivität verbunden und erlauben die gleichen Differenzierungen wie auch im Rahmen von Methoden (s. Bild 4-8) [OMG08-ol, S. 43ff].

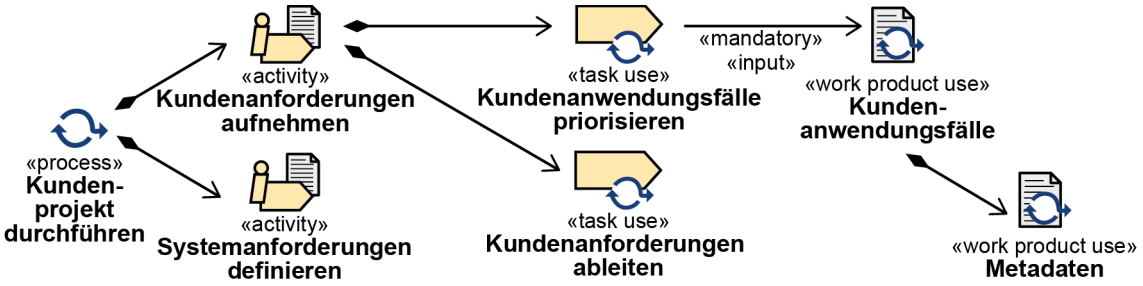


Bild 4-8: Mit SPEM definierter und hierarchisch strukturierter Prozess

- Durch das Paket **Prozessverhalten** (Process Behavior) ist es möglich, externe Ansätze zur Verhaltensmodellierung einzubinden, bspw. OMG BPMN. Das Paket definiert dazu, wie Prozesselemente eines eingebundenen Ansatzes zur Verhaltensmodellierung auf die Prozesselemente des Pakets für Prozessstrukturen abgebildet werden [OMG08-ol, S. 69].
- Für **Prozess- und Methodenkombinationen** (Process With Methods), also der Kombination von generischen, wiederverwendbaren Methoden und spezifischen, sequentiellen Prozessen, werden eigene Sprachkonstrukte bereitgestellt. Das Konzept wird in Bild 4-7-B deutlich. Neben den bereits bekannten spezifischen Rollen und Artefakten umfasst dies analog dazu *spezifische Aufgaben* (Task Use). Durch entsprechende Beziehungen (Content Trace) können spezifische Elemente mit generischen Methodenbestandteilen verknüpft werden. Um Prozesse und Methoden in der Organisation der Modelle trennen zu können, werden separate *Methodenpakete* (Method Content Package) und *Prozesspakete* (Process Package) bereitgestellt [OMG08-ol, S. 95ff].
- Für die gezielte (textuelle) **Dokumentation** (Managed Content) der modellierten Inhalte existiert ein eigenes Paket. Durch *Kategorien* (Categories) können Modellelemente nach eigenen Kriterien strukturiert werden. *Leitfäden* (Guidance) enthalten ergänzende Informationen zu Modellelementen, z. B. als Anleitung, Vorlage oder Checkliste. *Metriken* (Metrics) können Modellelementen zugeordnet werden, um diese messbar zu beschreiben (z. B. Aufwand einer Aktivität in Stunden). Für alle beschreibbaren Objekte in SPEM können strukturierte textuelle Beschreibungen (Kurz- und Langbeschreibung sowie Zweck) definiert werden [OMG08-ol, S. 73ff].
- Das Paket für **Bibliotheken** (Method Plugin) erlaubt es, Prozesse und Methoden zu modularisieren und zu erweitern. Dazu stehen als wesentliche Elemente *Methodenmodule* (Method Module), *Methodenkonfigurationen* (Method Configurations) und *Methodenbibliotheken* (Method Library) bereit, um die Modelle modular zu organisieren [OMG08-ol, S. 117ff].

Bewertung: SPEM stellt ein umfangreiches Metamodell zur allgemeinen Modellierung von Entwicklungsprozessen und -methoden dar. Dabei trennt der Ansatz strikt zwischen Prozess- und Methodenelementen. Es stehen umfangreiche Beschreibungsmittel bereit, um die Prozesse und Methoden zu strukturieren und ihre Abläufe zu spezifizieren. Im Vergleich zur BPMN ist SPEM wenig bekannt und intuitiv. Da SPEM ein Profil der

OMG UML darstellt, ist es sehr gut erweiterbar und findet breite Werkzeugunterstützung. SPEM gibt keine Methode oder ein Vorgehensmodell vor. Es bleibt damit unklar, welche Modellinhalte wie zu modellieren sind. Eine eindeutige Nutzung der Sprache ist daher ohne Modellierungsregeln kaum zu gewährleisten. Die Analyse von Anforderungen und Erarbeitung einer geeigneten Applikationsarchitektur werden nicht betrachtet, es wird davon ausgegangen, dass diese schon existiert und konkrete Aktivitäten unterstützen kann.

4.1.4 Spezifikation von Wertschöpfungssystemen nach SCHNEIDER

Die Dissertation von SCHNEIDER definiert eine Spezifikationstechnik zur Beschreibung und Analyse von Wertschöpfungssystemen. Ein Wertschöpfungssystem umfasst miteinander in Beziehung stehende Objekte zur Leistungserstellung mit dem Ziel der Befriedigung von Kundenbedürfnissen. Die Gestaltung eines Wertschöpfungssystems umfasst dabei die Marktleistungsentwicklung, -erstellung und -nutzung. Ziel der Spezifikationstechnik ist die Planung von Wertschöpfungssystemen mit Hilfe eines interdisziplinären Kooperations- und Kommunikationsmittels und eines systematischen Vorgehens.

Die Spezifikationstechnik besteht aus einer graphischen **Modellierungssprache** mit einem Sprachkonzept, welches im Kern durch ein Metamodell (abstrakte Syntax und statische Semantik) sowie einer grafischen Notation (konkrete Syntax) beschrieben wird. Die Modellierungssprache nutzt zur Beschreibung der Modelle vier Arten von Sprachkonstrukten: Grund- und Zusatzkonstrukte, Beziehungen und Verweise. Weiterhin wird die Bildung entsprechender Sichten erlaubt. Die abbildbaren sieben kohärenten, vernetzten Partialmodelle sind in Bild 4-9 dargestellt. Sie ordnen sich in drei Ebenen ein [Sch18, S. 114ff].

- Die **strategische Ebene** beschreibt im *Geschäftsmodell* als Resultat der strategischen Produktplanung ein aggregiertes Abbild der Geschäftslogik eines Unternehmens („Was?“). Es ist nach einer Business Model Canvas in Angebots-, Kunden-, Finanz-, Absatz- und Wertschöpfungsmodell strukturiert. Letzteres beschreibt die Schlüsselaktivitäten, -ressourcen und -partner sowie die Wertschöpfungsstruktur.
- Die notwendigen Kompetenzen und Anforderungen zur Leistungserstellung („Womit?“) werden in der **taktischen Ebene** beschrieben. Sie werden in den vier Partialmodellen *Anforderungen* (Liste attributierter Anforderungen), *Aktivitäten* (Hierarchie der Geschäftsprozesse gegliedert nach Funktionsbereichen), *Ressourcen* (Hierarchie der Personal- und Sachmittel) und der *Aufbauorganisation* (Rollen in den Funktionsbereichen und Zuordnung zu Aktivitäten und Ressourcen) abgebildet.
- Die **operative Ebene** umfasst die konkrete Umsetzung der Leistungserstellung („Wie?“) in zwei Partialmodellen. Das *Interaktionsmodell* definiert die Akteure des Wertschöpfungssystems als Black-Box und setzt diese über Marktleistungs-, Informations- und Geldflussbeziehungen in Form eines Blockdiagramms in Relation. Die *Ablauforganisation* beschreibt die Struktur der Akteure, indem sie Aktivitäten auf die Aufbauorganisation abbildet und Ressourcen zuordnet. Es erweitert dazu die Sprache OMEGA.

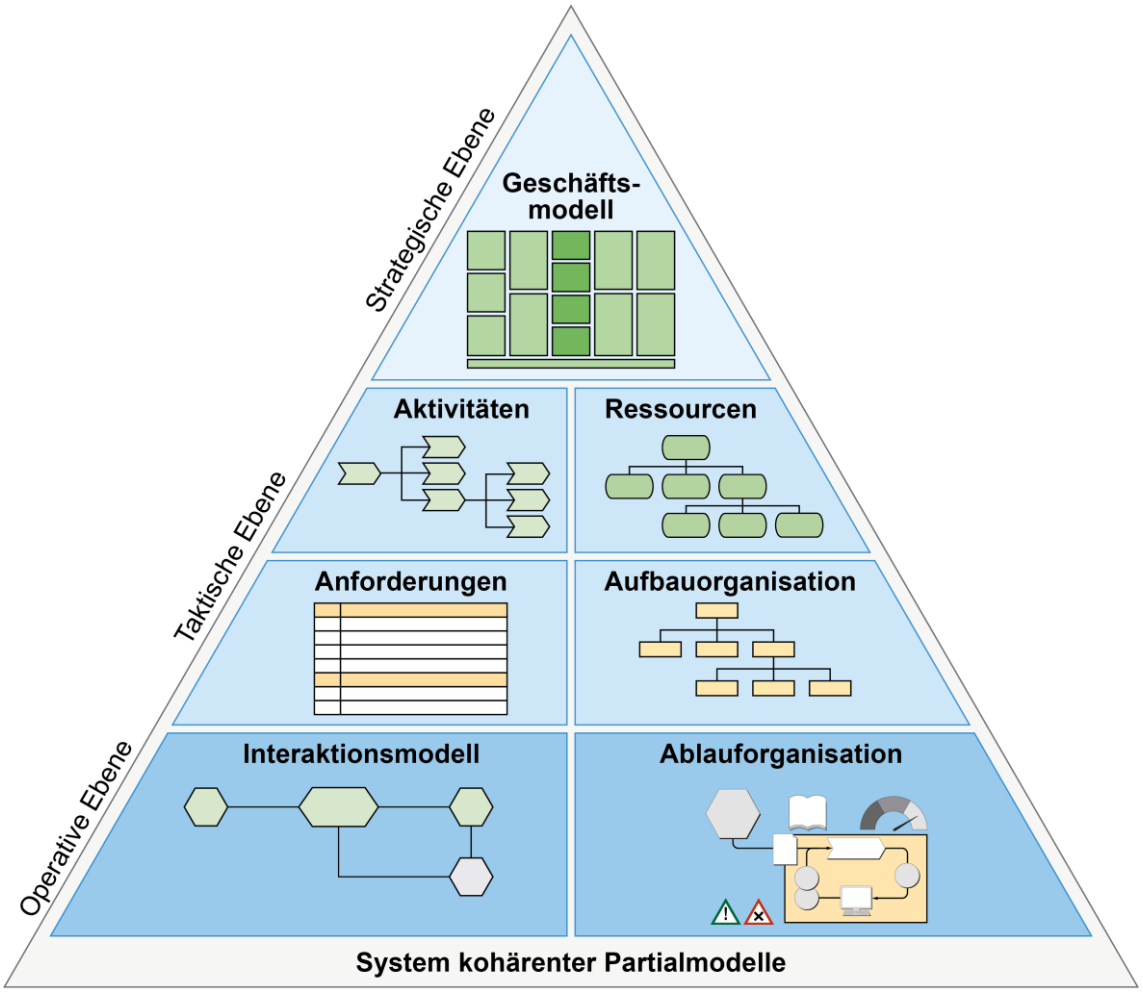


Bild 4-9: Die sieben Partialmodelle zur Beschreibung und Analyse von Wertschöpfungssystemen nach SCHNEIDER *[Sch18, S. 114]*

Als **Werkzeugunterstützung** wird ein Kartenset für Workshops mit der entsprechenden konkreten Syntax definiert. Darüber hinaus wurden prototypische Umsetzungen in zwei Softwarelösungen implementiert. Das **Vorgehensmodell** beschreibt die durchzuführenden Aufgaben und definiert die Nutzung der Partialmodelle und der Werkzeuge. Es berücksichtigt dazu verschiedene Startpunkte zum Aufbau der Modelle.

Bewertung: Die Spezifikationstechnik ist einleuchtend in die Bestandteile Modellierungssprache, Werkzeugunterstützung und Vorgehensmodell strukturiert und enthält somit alle wesentlichen Bestandteile zur Modellierung. Sie beschreibt wesentliche Aspekte von Wertschöpfungssystemen und bezieht dabei auch die Produktentstehung mit ein. Zwar erlaubt die Spezifikationstechnik eine erste Analyse und Spezifikation von Prozessen, Ressourcen, sowie Anforderungen und kann damit zur Geschäftsarchitektur beitragen, allerdings ist die Abstraktionsebene dabei zu grob für die Gestaltung der Engineering-IT. Methoden werden zwar identifiziert, jedoch nicht weiter detailliert. Die systematische Ableitung bzw. Erstellung einer Daten- und Applikationsarchitektur werden nicht berücksichtigt.

4.2 Ansätze zur Analyse von Anforderungen und Prozessen

Es exisitieren verschiedene Ansätze zur systematischen Identifikation und Analyse von Anforderungen. Die hier vorgestellten Ansätze beziehen sich auf technische Systeme oder auf Prozesse. Im Folgenden werden daher Ansätze eingeschlossen, die einen methodischen Beitrag zur Anforderungsanalyse leisten und die Analyse der Prozessebene unterstützen.

4.2.1 Ansätze zum Requirements Engineering Technischer Systeme

Zur Erhebung von Anforderungen im Rahmen der Entwicklung technischer Systeme existieren verschiedene Ansätze. Diese Ansätze ähneln sich und sind sehr umfangreich, sodass sie im Folgenden zusammengefasst beschrieben werden. Das **Requirements Engineering** verfolgt folgende Ziele [RS14, S. 13].

- *Anforderungen kennen:* Relevante Anforderungen identifizieren und unter den Stakeholdern einen Konsens zu diesen bilden. Die Anforderungen müssen nach vorgegebenen Richtlinien dokumentiert und verwaltet werden.
- *Bedürfnisse verstehen:* Die Wünsche und Bedürfnisse von Stakeholdern verstehen und festhalten.
- *Risiko minimieren:* Das Risiko, ein System zu entwickeln, welches nicht den Bedürfnissen der Stakeholder entspricht, muss durch geeignete Spezifikation und Verwaltung der Anforderungen minimiert werden.

Die ISO/IEC/IEEE 29148 definiert eine **Anforderung** als „statement which translates or expresses a need and its associated constraints [...] and conditions“. Sie beschreiben ein System, Software oder Service. Anforderungen drücken einen Bedarf aus und können auf verschiedenen Ebenen eines Systems existieren [ISO29148, S. 4f, 12f]. Anforderungen müssen Qualitätskriterien genügen, welche bspw. in [ISO29148, S. 12f] und [RWZ+17, S. 19ff] festgelegt sind. Darunter fallen für einzelne Anforderungen die Notwendigkeit, Angemessenheit des Detaillierungsgrades, Eindeutigkeit, Vollständigkeit für ihr Verständnis, Singularität ihres Beschreibungsgegenstandes, die Umsetzbarkeit, Verifizierbarkeit, Korrektheit und Konformität mit anzuwendenden Regeln zur Dokumentation. Eine Menge von Anforderungen muss die Vollständigkeit für ihr Verständnis, die Konsistenz zwischen den einzelnen Anforderungen, ihre Umsetzbarkeit, Verständlichkeit und Validierbarkeit aufweisen.

POHL sowie EBERT unterscheiden drei **Arten** von Anforderungen. *Funktionale Anforderungen* beschreiben eine Funktionalität eines Systems und dadurch, was das System tun soll. Beispiele für funktionale Anforderungen sind Funktionen, Ablaufbeschreibungen oder Daten- und Schnittstellenanforderungen. *Qualitätsanforderungen* beschreiben qualitative Eigenschaften eines Systems, die nicht durch funktionale Anforderungen abgedeckt werden. Beispiele sind Performanz, Benutzerfreundlichkeit oder Sicherheit des Systems. *Randbedingungen* sind Anforderungen, die den Lösungsraum einschränken. Sie können

im Rahmen der Entwicklung eines Systems weder unmittelbar umgesetzt noch beeinflusst werden und müssen als gegeben hingenommen werden. Neben den drei genannten Arten gibt es weitere kontextspezifische Arten, die diese elementaren Arten weiter untergliedern [PR15, S. 8f], [Ebe19, S. 29ff].

Anforderungen können über verschiedene **Ebenen** abstrahiert werden, wobei die übergeordnete Ebene die Problemstellung für die untergeordnete Ebene konkretisiert und den Lösungsraum einschränkt. Innerhalb dieser Ebenen beschreiben Anforderungen, *Was* vom System erwartet wird. Auf deren Basis können Architekturen erarbeitet werden, die beschreiben, *Wie* die Anforderungen umgesetzt werden [Ebe19, S. 23], [HJD11, S. 26]. Die Anforderungen der untergeordneten Ebene basieren wiederum auf den zuvor getroffenen Architekturentscheidungen [ISO29148, S. 17f], [Wei14, S. 182]. Dadurch kann sowohl der Problem- als auch der Lösungsraum inkrementell und parallel konkretisiert werden. Es wird ersichtlich, dass RE nach diesem Verständnis keine einzelne Phase ist, sondern kontinuierlich und phasenübergreifend durchgeführt werden muss [PR15, S. 5]. Dieses Vorgehen kommt in dem in Bild 4-10 dargestellten Twin-Peak Modell zum Ausdruck. Die Ebenen können dabei z. B. durch Hierarchiestufen des Systems, aber auch verschiedene Betrachtungsweisen wie funktionale, logische (Lösungsansätze und -prinzipien) oder physische (konkrete, implementierbare Elemente) Ebenen gebildet werden [WLR+15, S. 92ff], [MFH15, S. 1ff].

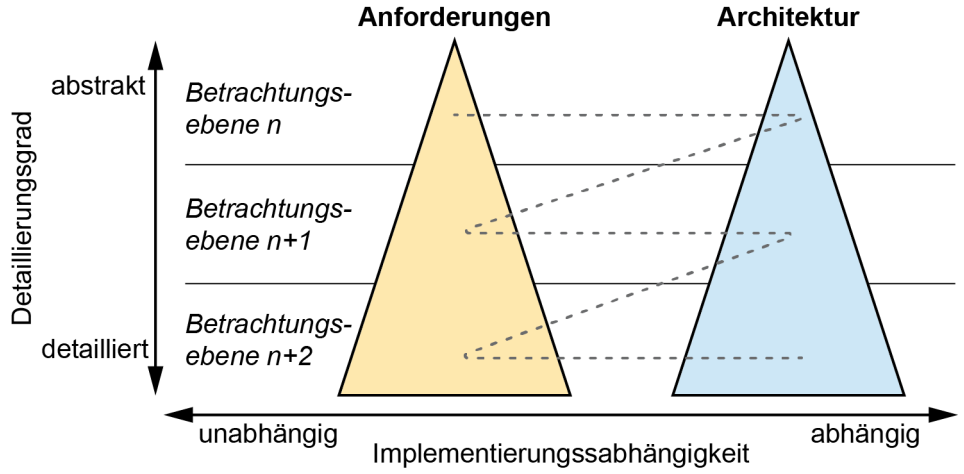


Bild 4-10: Das Twin-Peaks Modell zur wechselseitigen Ableitung und Detaillierung der Anforderungen und Architektur in Anlehnung an [Nus01, S. 116]

Um die Anforderungen einer Ebene zu erarbeiten, existieren verschiedene Ansätze, welche Kernaktivitäten des Requirements Engineering definieren, bspw. nach [RS14, S. 14], [Ebe19, S. 33ff], [PR15, S. 4f] oder [ISO29148, S. 9]. Die Ansätze weisen ähnliche Aktivitäten auf, die durch Prozesse (bspw. aus der ISO/IEC/IEEE 15288 [ISO15288] oder ISO/IEC/IEEE 29148 [ISO29148]) in eine Reihenfolge gebracht werden können. Im Folgenden werden die in Bild 4-11 dargestellten Aktivitäten nach der Gliederung von POHL UND RUPP beschrieben [PR15, S. 12]. Die Aktivitäten der anderen Ansätze sind ähnlich definiert und können grob aufeinander abgebildet werden.

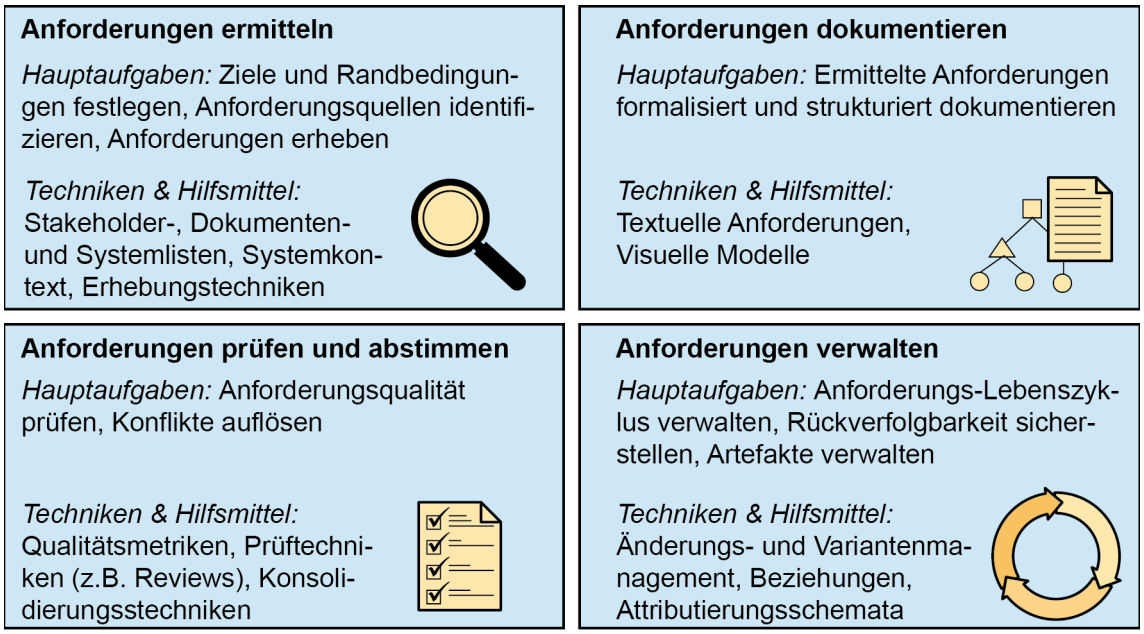


Bild 4-11: Die vier Kernaktivitäten des Requirements Engineering

- **Anforderungen ermitteln:** Zunächst werden der Untersuchungsgegenstand, Ziele und Randbedingungen festgelegt [Ebe19, S. 53ff, 72ff, 94ff]. Anforderungsquellen werden systematisch identifiziert, z. B. mit Hilfe von Stakeholder-, Dokumenten- und Systemlisten [RS14, S. 77f]. Auch in Kontextbeschreibungen können Anforderungsquellen ausgemacht werden, z. B. Schnittstellen zu Systemen und Akteuren [GDE+18, S. 419]. Die so identifizierten Anforderungsquellen werden systematisch abgearbeitet, um die Anforderungen zu erheben. Dazu werden Ermittlungstechniken wie z. B. artefaktbasierte, Kreativitäts-, Beobachtungs- oder Befragungstechniken genutzt [RS14, S. 98ff], [PR15, S. 32ff].
- **Anforderungen dokumentieren:** Die ermittelten Anforderungen müssen formalisiert und strukturiert beschrieben werden, um die Kommunikation mit allen Beteiligten zu ermöglichen [Ebe19, S. 105]. Dazu bieten sich zum einen textuelle Anforderungen an, die natürliche Sprache nutzen und mittels Satzschablonen, welche bestimmte Satzstrukturen definieren, formalisiert werden [RS14, S. 216ff], [RWZ+17, S. 35ff]. Zum anderen können semi-formale Systemmodelle genutzt werden, die bspw. Prozesse, Abläufe, Strukturen oder Anwendungsfälle abbilden [Ebe19, S. 144ff], [HJD11, S. 48ff]. Sammlungen von Anforderungen, sogenannte Spezifikationen, werden zur besseren Verständlichkeit in hierarchische Strukturen gegliedert, z. B. nach der ISO/IEC/IEEE 29148 [ISO29148, S. 51ff]. Zudem kann ein Glossar erstellt werden, um verwendete Begriffe eindeutig zu definieren [Ebe19, S. 128f].
- **Anforderungen prüfen und abstimmen:** Anforderungen müssen auf die Einhaltung der bereits vorgestellten Qualitätskriterien geprüft werden [PR15, S. 97ff]. Dazu können verschiedene Prüftechniken genutzt werden, bspw. Reviews oder die Erstellung von Prototypen oder Testfällen [RS14, S. 318ff], [Ebe19, S. 186ff]. Da es zu wider-

sprüchlichen Anforderungen innerhalb einer Spezifikation kommen kann, werden Konflikte im Rahmen der Anforderungsabstimmung gezielt identifiziert, analysiert und aufgelöst [PR15, S. 115ff], [RS14, S. 347ff].

- **Anforderungen verwalten:** Anforderungen unterliegen einem Lebenszyklus, der im Rahmen des Requirements Management (RM) verwaltet wird. Ein wesentlicher Punkt ist daher das Änderungsmanagement von Anforderungen, welches Prozesse zur Versionierung, zur Nachverfolgung von Änderungen an Anforderungen sowie die Analyse ihrer Auswirkungen umfasst [Ebe19, S. 249ff], [PR15, S. 140ff]. Um die Entwicklung der Anforderungen über den gesamten Lebenszyklus des Systems sowie die Zusammenhänge der Anforderungen nachvollziehen zu können, ist im Rahmen des RM eine Vor- und Rückverfolgbarkeit umzusetzen. Die Anforderungen sind in den in Bild 4-10 dargestellten Ebenen dazu horizontal als auch vertikal mit anderen Artefakten verknüpft [Ebe19, S. 255ff], [RS14, S. 421ff]. Zum RM gehört auch die Definition verschiedener Artefakttypen (z. B. Anforderungen, Testfälle), ihrer Zustände und Attribute [RS14, S. 376f], [PR15, S. 123ff].

Bewertung: Das RE unterstützt gezielt mit Hilfe von Methoden die schrittweise Erarbeitung und Verwaltung von Anforderungen. Dazu steht eine Vielzahl von Techniken bereit, die situationsspezifisch genutzt werden müssen. In diesem Rahmen werden auch Modelle berücksichtigt, die Geschäftsprozesse oder Datenstrukturen abbilden. Es wird jedoch kein Vorgehen beschrieben, wie diese Modelle schrittweise aufgebaut werden können. Die für die Modelle zu nutzenden Sprachen werden typischerweise nicht durch das RE, sondern durch geeignete Beschreibungsansätze definiert (vgl. Kap. 4.1). Grundsätzlich werden eher technische Systeme als die Engineering-IT als Gestaltungsgegenstand betrachtet. Durch die Formalisierung der Anforderungen kann von einem Modellierungsrahmenwerk gesprochen werden, welches jedoch nicht alle notwendigen Aspekte zur Gestaltung der Engineering-IT umfasst. So werden die Gestaltung und Evaluierung von Applikationsarchitekturen nicht durch das RE berücksichtigt.

4.2.2 Geschäftsprozessmanagement nach SCHMELZER UND SESSELMANN

Zum Management von Geschäftsprozessen existieren zahlreiche Publikationen verschiedener Autoren, wobei die grundsätzliche Struktur im Folgenden an SCHMELZER UND SESSELMANN angelehnt ist. Das Geschäftsprozessmanagement (GPM) ist ein integriertes System, bestehend aus Führung, Organisation und Controlling. Es verfolgt den Zweck, Geschäftsprozesse zielgerichtet zu steuern und zu optimieren. Integriert bedeutet in diesem Kontext, dass Aufgaben, Hilfsmittel und IT-Systeme mit in die Betrachtung einbezogen werden [SS13, S. 6], [Gad17, S. 19ff]. GPM umfasst die Dokumentation, Analyse und Restrukturierung von Arbeitsabläufen [Gad17, S. 1]. Bild 4-12 stellt die Aufgabenfelder des GPM zusammenhängend dar.

Die **Prozessführung** fokussiert die Führung der in den Geschäftsprozessen tätigen Menschen, damit sie auf die Ziele und Orientierungen des GPMs hinarbeiten [SS13, S. 9].

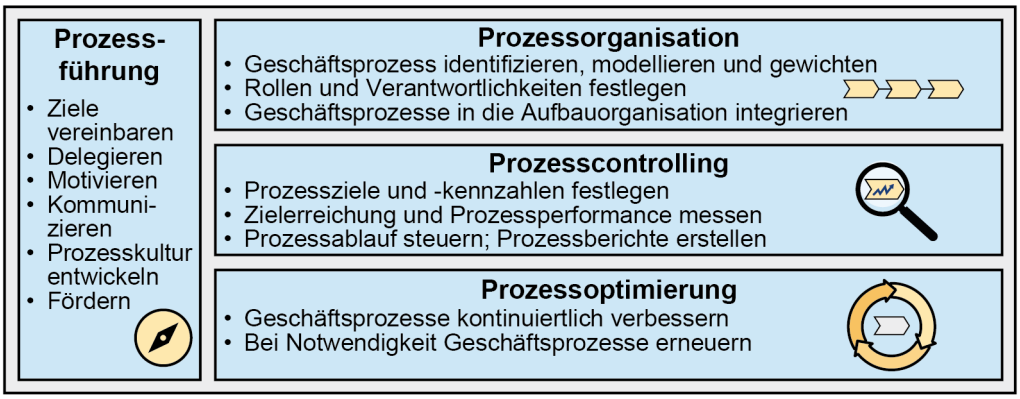


Bild 4-12: Aufgabenfelder des Geschäftsprozessmanagement in Anlehnung an [SS13, S. 9]

Die **Prozessorganisation** umfasst alle Aufgaben zur Identifizierung und Gestaltung der Geschäftsprozesse sowie die Integration in die Aufbauorganisation. Nach der Identifikation und Abgrenzung relevanter Prozesse werden diese aus fachlich-konzeptioneller Sicht modelliert. Dazu werden entsprechende Modellierungssprachen (s. Kap. 4.1.3) genutzt und die Prozessabläufe entsprechend den strategischen Zielen einer Organisation umgestaltet [Gad17, S. 21], [SL12, S. 167ff], [SS12, S. 195ff]. Hierfür kann auf Restrukturierungsansätze zurückgegriffen werden, z. B. das Weglassen, Zusammenfassen oder Parallelisieren von Aktivitäten [Gad17, S. 34f]. Im Rahmen des **Prozesscontrollings** werden anhand von Prozesskennzahlen die Prozessziele geplant und die Zielerreichung gesteuert. Besondere Bedeutung kommt der **Prozessoptimierung** zu. Diese zielt auf eine dauerhafte Erhöhung der Effektivität und Effizienz von Unternehmensprozessen ab [SS13, S. 10].

Bewertung: Das Geschäftsprozessmanagement unterstützt bei der Abbildung, Restrukturierung und Analyse von Prozessen. Der Fokus liegt dabei auf den Aktivitäten und ihrer sachlogischen Abfolge, die entstehenden Artefakte und notwendigen IT-Systeme werden nur am Rande betrachtet. Es gibt keine unmittelbaren methodischen Hilfestellungen zur Analyse von Anforderungen, die aus Prozessen resultieren. Eine explizite Berücksichtigung und Abbildung von Methoden erfolgt nicht.

4.2.3 Model-Based Process Engineering nach HUTH ET AL.

Der Ansatz des Model-Based Process Engineering (MBPE) nach HUTH ET AL. ermöglicht eine integrierte Sicht auf die Geschäftsprozesse und Artefakte im Engineering. Dazu wird ein Erklärungs- und Handlungsmodell für die Implementierung von SE Ansätzen bereitgestellt. Die in Bild 4-13 dargestellten Partialmodelle dienen als Grundlage [HIW+18, S. 7].

Das **Produktmodell** (Product System Model) umfasst alle Modelle, die der Beschreibung des zu entwickelnden technischen Systems dienen. Diese können in Partialmodelle gegliedert sein. Darunter können bspw. Anforderungs- oder Funktionsmodelle fallen. Dazu werden ihr Inhalt, die Struktur sowie die Beziehungen der Partialmodelle des Produktmodells untereinander beschrieben. Die Partialmodelle werden zudem mit den Aktivitäten des Prozessmodells verknüpft [HIW+18, S. 7].

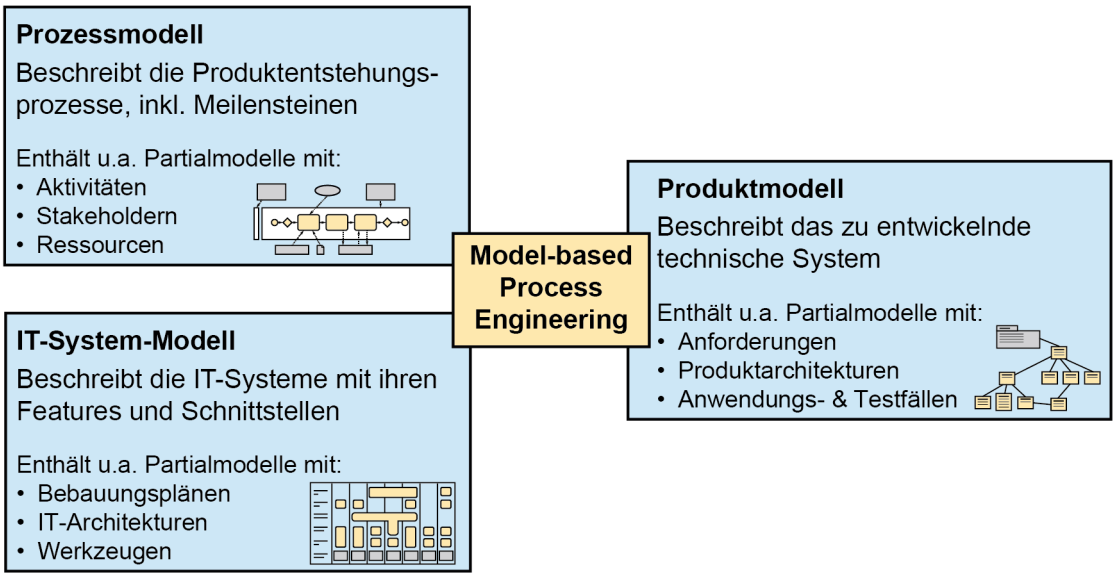


Bild 4-13: Die drei Partialmodelle des MBPE in Anlehnung an [HIW+18, S. 7]

Das **Prozessmodell** (Process Model) beschreibt notwendige Aktivitäten zur Entwicklung eines Produkts. In diesem Rahmen werden Verantwortlichkeiten, Entscheidungspunkte und Ressourcen definiert. Die Beschreibung der Prozesse kann dabei verschiedene Abstraktionsebenen, von der Übersichts- bis zur Ausführungsebene, umfassen. Im Rahmen der Implementierung von SE dient das Modell dazu, Schwachstellen der bisherigen Arbeitsweisen und Vorteile neuer Arbeitsweisen zu identifizieren [HIW+18, S. 8].

Mit dem **IT-System Modell** (IT System Model) soll die gezielte Entwicklung der IT Infrastruktur ermöglicht werden. Die IT-Werkzeuge sollen dazu innerhalb einer geeigneten IT-Architektur die sich aus dem Produkt- und Prozessmodell ergebenden Anforderungen umsetzen. Als Beispiele für mögliche Inhalte werden Bebauungspläne und Systemarchitekturen genannt [HIW+18, S. 7f].

Zur **Modellierung** des Prozessmodells wird die Modellierungssprache OMG BPMN genutzt, für die weiteren Modellelemente ein Profil der OMG UML bzw. SysML (vgl. Kap. 4.1.3 und 4.1.2). Dies erlaubt es, die Elemente der drei MBPE-Partialmodelle miteinander zu verknüpfen. Zur Erstellung der Modelle wird ein **Vorgehensmodell** vorgestellt, welches in Bild 4-14 dargestellt ist [HIW+18, S. 7]. Das Vorgehen erfolgt wie in Bild 4-14 dargestellt. Dazu wird in einem ersten Schritt die vorhandene Prozessdokumentation für einen SOLL-Prozess zusammengetragen und gesichtet. Diese dient dann als Grundlage, um in der folgenden Phase den Prozess mit Hilfe des bereitgestellten Sprachprofils zu modellieren. Bereits bekannte Bestandteile des Produktmodells werden zunächst als Platzhalter modelliert und mit den Elementen des Prozessmodells verknüpft. In einer dritten Phase wird das Modell im Rahmen von Workshops mit den Stakeholdern und Ownern der betroffenen Prozesse gereviewt. Dabei werden Abweichungen zum tatsächlich gelebten Prozess sowie Produktmodell dokumentiert. In der letzten Phase wird der Prozess vervollständigt. Anschließend werden die Platzhalter des Produktmodells ausgearbeitet [HIW+18, S. 9ff, 12f].

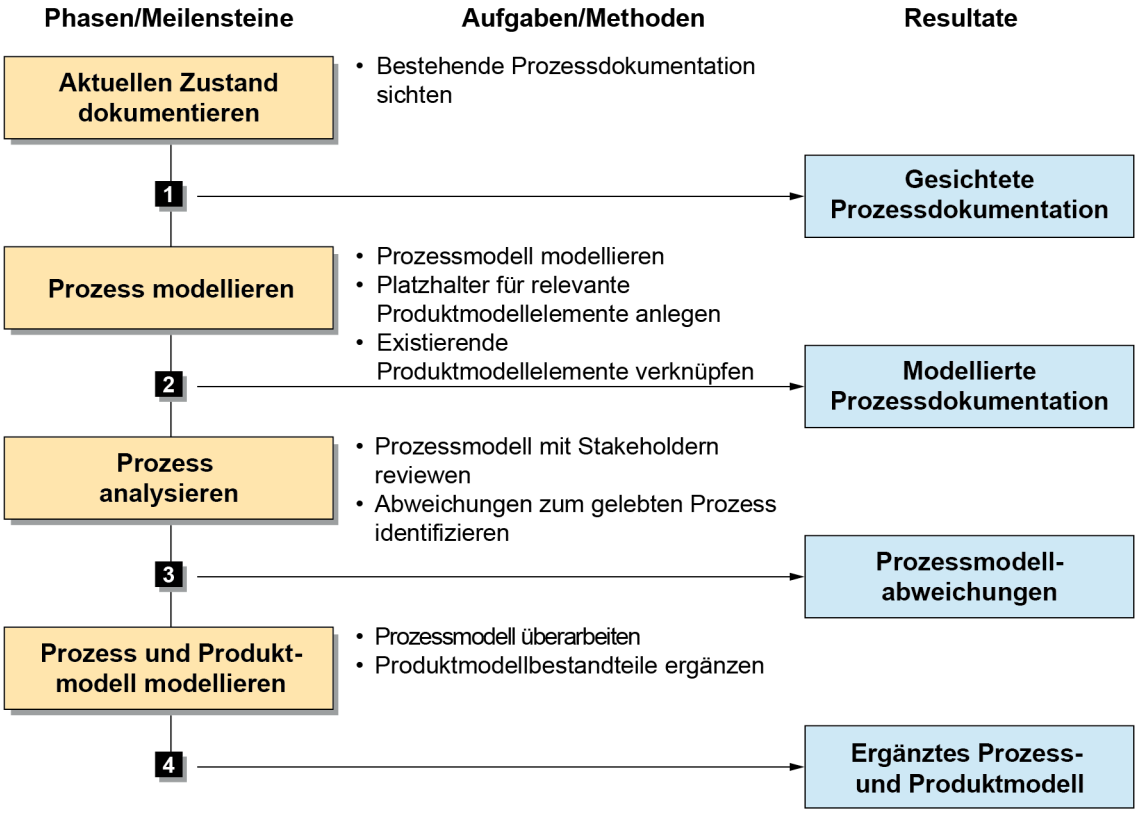


Bild 4-14: Vorgehensmodell des MBPE nach HUTH ET AL.

Bewertung: Der Ansatz betrachtet explizit das Themenfeld des Engineerings und umfasst durch die drei beschriebenen Partialmodelle wesentliche Bestandteile einer Geschäfts-, Prozess- und Applikationsarchitektur. Eine entsprechende Modellierungssprache mit Werkzeugunterstützung durch Werkzeuge zur Systemmodellierung wird bereitgestellt. Für die Erstellung des Prozess- und Produktmodells wird ein entsprechendes Vorgehen dargelegt. Dieses geht jedoch nicht auf das IT-System Modell ein und umfasst somit nicht die Gestaltung und Evaluierung von Applikationsarchitekturen. Es wird keine Methode zur angeleiteten Erstellung der Modelle vorgegeben. Die Ableitung von Anforderungen und ihre Verwaltung sowie die Modellierung von Methoden werden nicht berücksichtigt.

4.3 Ansätze zur Definition von Enterprise und IT-Architekturen

In Kapitel 3.4.8 wurde bereits das grundlegende Konzept von Enterprise Architekturen beschrieben. Nachfolgend soll ein repräsentativer Ansatz detaillierter vorgestellt werden. Darüber hinaus werden spezifische Ansätze betrachtet, welche die Spezifikation von IT-Architekturen im Engineering systematisch unterstützen.

4.3.1 The Open Group Architecture Framework (TOGAF)

Wie bereits in Kapitel 3.4.8 dargelegt, greifen EAM-Ansätze die Idee auf, eine Organisation ganzheitlich zu betrachten und zu modellieren. Es existiert eine Vielzahl von

EAM-Ansätzen. MATTHES gibt in [Mat11, S. 37ff] einen umfassenden Überblick über verfügbare EAM-Ansätze. HERDEN UND ZENNER stellen in [HZ11, S. 5ff] einen Klassifikationsansatz für EAM-Ansätze vor. Sie kommen zu dem Schluss, dass TOGAF einer der umfassendsten EAM-Ansätze ist und daher als Stellvertreter für EAM-Ansätze betrachtet werden kann [HZ11, S. 17], weshalb dieser Ansatz im Rahmen dieses Kapitels exemplarisch vorgestellt werden soll. The Open Group Architecture Framework (TOGAF) ist das aktuell am weitesten verbreitete und bekannteste EA Framework [Han16, S. 29] und damit zu einem De-Facto-Standard geworden [Kel17, S. 305f]. Es wird von der Open Group spezifiziert, einem industriellen Konsortium, welches technologische Standards definiert. Wesentlicher Bestandteil ist die in Bild 4-15 dargestellte **Architecture Development Method (ADM)**, welche in acht von A bis H numerierten Phasen (und einem vorbereitenden Schritt) die Erstellung einer EA beschreibt.

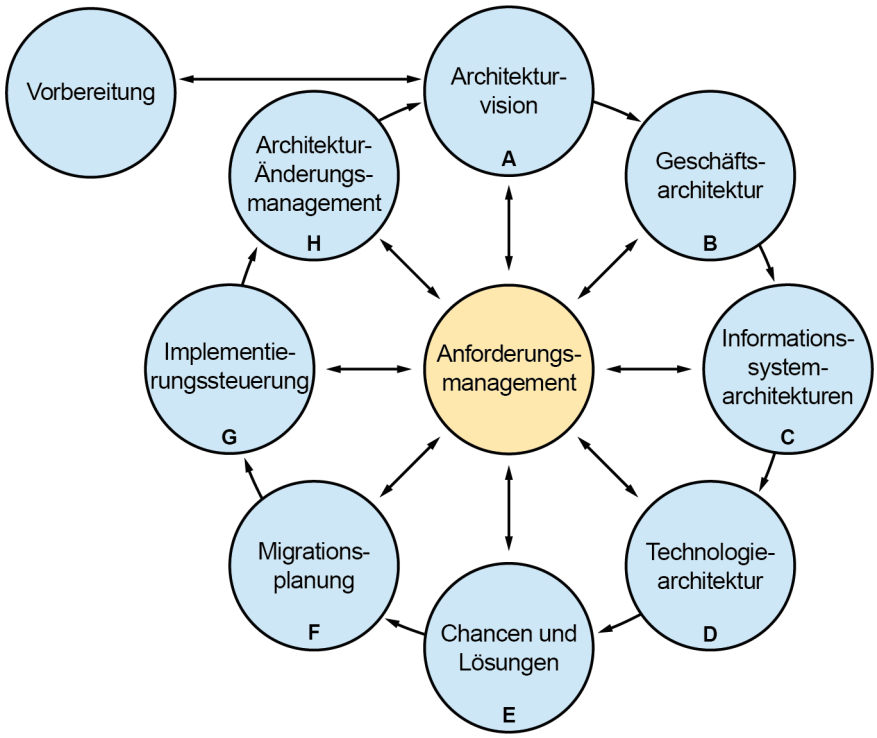


Bild 4-15: Die acht Schritte zur Erstellung einer Unternehmensarchitektur mit TOGAF in Anlehnung an [TOG18, S. 40]

- **Vorbereitung:** Es werden grundlegende vorbereitende Schritte beschrieben. Da TOGAF einen modularen und generischen Ansatz verfolgt, muss es zudem an die Bedürfnisse und Gegebenheiten angepasst werden. Dazu gehört auch die Integration mit anderen Frameworks für Bereiche, die TOGAF nicht abdeckt, z. B. für die Lösungsentwicklung. Darüber hinaus werden IT-Werkzeuge festgelegt und Architekturprinzipien definiert [TOG18, S. 53ff].
- **Architekturvision (A):** Zunächst wird ein Projekt aufgesetzt und und der Betrachtungsgegenstand und Rahmenbedingungen sowie Ziele definiert. Stakeholder, ihre Concerns und Geschäftsanforderungen werden identifiziert. Eine Architekturvision

stellt ein Zielbild für die Hauptstakeholder bereit, in dem sie strategische Grundentscheidungen und eine übergreifende Architektur mit hohem Abstraktionsgrad für den IST- und SOLL-Zustand festlegt [TOG18, S. 67ff].

- **Architekturen (B-D):** In den Phasen B bis D wird in einem jeweils analogen Vorgehen die *Geschäftsarchitektur*, die *Informationssystemarchitektur* (bestehend aus Daten- und Applikationsarchitektur) und die *Technologiearchitektur* definiert (vgl. Kap. 3.4.8). Im Wesentlichen umfasst dies die Entwicklung einer IST- und SOLL-Architektur und einer darauf basierenden Defizitanalyse. Unterstützend werden eine Roadmap definiert und unerwünschte Wechselwirkungen zwischen den Architekturebenen aufgelöst [TOG18, S. 78ff, 97ff, 109ff, 120ff].
- **Chancen & Lösungen (E):** Zur Implementierung der definierten SOLL-Architekturen werden die bisherigen Ergebnisse konsolidiert und Umsetzungsmöglichkeiten wie bspw. Projekte identifiziert. Abhängigkeiten werden identifiziert und Schnittstellenanforderungen abgestimmt. Im Rahmen einer Implementierungsstrategie werden Arbeitspakete, Übergangsarchitekturen und Roadmaps definiert [TOG18, S. 133ff].
- **Migrationsplanung (F):** Die Migrationsplanung beschreibt, wie man durch die Ausführung der Implementierungsstrategie von der IST- zur SOLL-Architektur kommt. Dazu werden den Arbeitspaketen Business Values zugeordnet, benötigte Ressourcen definiert und Migrationsprojekte definiert und priorisiert und in eine Roadmap überführt [TOG18, S. 141ff].
- **Implementierungssteuerung (G):** Die Implementierungsmaßnahmen werden bis zum Abschluss aller Maßnahmen begleitet, um bei Abweichungen zur vereinbarten SOLL-Architektur gegensteuern zu können [TOG18, S. 150ff].
- **Architektur-Änderungsmanagement (H):** Durch eine ständiges Monitoring soll sichergestellt werden, dass die Architektur den Bedürfnissen und Anforderungen der Organisation entspricht. Das Änderungsmanagement ermöglicht, Anpassungen aufgrund technologischer oder geschäftlicher Änderungen an der Architektur vorzunehmen und die sich ergebenden Auswirkungen zu analysieren und umzusetzen [TOG18, S. 157ff].
- **Anforderungsmanagement:** Architekturanforderungen werden kontinuierlich in allen relevanten Phasen identifiziert und über den weiteren Lebenszyklus verwaltet. Zur Identifikation werden geeignete Techniken wie bspw. Geschäftsszenarien in den jeweiligen Phasen genutzt. Zur Anforderungsverwaltung wird ein Prozess vorgeben, der insbesondere die Änderungen über den Lebenszyklus berücksichtigt [TOG18, S. 166ff].

Die **ADM Guidelines and Techniques** sind eine Sammlung von Richtlinien und Techniken (z. B. Stakeholder Management), die ausgewählt werden können, um die jeweiligen Schritte der ADM situationsspezifisch zu unterstützen. Das **Architecture Content Framework** beschreibt, wie die in der ADM erzeugten Artefakte konsistent definiert, strukturiert

und präsentiert werden können. Dies wird durch ein Metamodell unterstützt, bestehend aus einem Kern und sechs modularen Erweiterungen zur detaillierteren Modellierung von Services, Prozessen, Daten, Motivation, operativer Steuerung und Konsolidierung der Infrastruktur. Weiterhin wird die Semantik der Konstrukte textuell beschrieben. Dabei orientiert sich TOGAF auch an den Normen ISO/IEC/IEEE 42010 [ISO42010] und ISO/IEC/IEEE 15288 [ISO15288]. Die **Enterprise Continuum & Tools** beschreiben mit dem dreigliedrigen Enterprise Continuum ein Klassifizierungsschema für Kontext-, Architektur- und Implementierungselemente. Darüber hinaus wird beschrieben, wie Architekturen zur kollaborativen Arbeit und Wiederverwendung partitioniert und in einem Repository verwaltet werden können. Das **Architecture Capability Framework** beschreibt Organisationsstrukturen, Prozesse, Rollen und Verantwortlichkeiten, um die Fähigkeiten einer Organisation für effektives EAM zu etablieren [Krc15, S. 627f], [TOG18, S. 175, 271ff, 373, 405]

Bewertung: TOAGF ist extrem umfassend und stellt ein Metamodell zur Abbildung zahlreicher Aspekte einer Organisation bereit. Eine schrittweise Methode zur Erstellung einer EA unterstützt dabei. Aufgrund der generischen Ausrichtung und des hohen Abstraktionsniveaus von TOGAF werden Spezifika des Engineerings nicht berücksichtigt. Es werden lediglich generische Hilfsmittel und Diagramme zur Visualisierung vorgeschlagen. Vielmehr erfordert TOGAF eine umfassende Anpassung an den Einsatzzweck, welche aufgrund des Umfangs komplex ist [Krc15, S. 629]. Die Analyse von Methoden im Engineering wird nicht explizit unterstützt. Da TOGAF kein Framework zur Lösungsfindung darstellt, wird die Definition geeigneter Architekturen und die Auswahl konkreter IT-Werkzeuge nicht methodisch unterstützt. Eine Sprachkonzept zur Modellierung der Architekturelemente wird nicht bereitgestellt.

4.3.2 Informationssystem-Architekturen nach HOFFMANN UND HEIMES

HOFFMANN UND HEIMES fokussieren mit ihrem Ansatz produzierende Unternehmen, welche Bedarf an durchgängigen, IT-unterstützten Prozessen besitzen. Sie zeigen dazu eine mehrdimensionale Morphologie zur Beschreibung von Informationsystemen (IS) auf. Unter Nutzung der Morphologie-Merkmale wird ein multidimensionaler Architekturraum aufgespannt und Gestaltungsoptionen für die Digitalisierung eines Unternehmens gegeben. Letzlich soll ein Unternehmen durch eine geeignete Zielarchitektur befähigt werden, durchgängige Prozesse mit IT zu unterstützen [HH18, S. 984f].

Die **Morphologie** beschreibt eine Logik zur Integration von Informationssystemen in vier Dimensionen. *Technische Merkmale* beschreiben informationstechnische Eigenschaften, z. B. die technische Umsetzung von Schnittstellen. *Organisatorische Merkmale* betreffen die Menschen, die mit dem Informationssystem in seinen Lebenszyklusphasen interagieren. *Prozessuale Merkmale* weisen einen Bezug zu den Geschäftsprozessen des Unternehmens auf. Die *Daten betreffende Merkmale* bilden die im Informationssystem gespeicherten Daten unter Berücksichtigung von Qualität und Struktur ab [HH18, S. 990]. Eine beispielhafte Ausprägung innerhalb der Morphologie stellt unter den technischen Merkmalen die

Kategorie „Schnittstellen“ dar, die mit den Ausprägungen „generisch“ oder „spezifisch“ charakterisiert werden.

Basierend auf den vier Dimensionen werden die in Bild 4-16 gezeigten **Darstellungsformen** zur Beantwortung von Integrationsfragen vorgestellt. Die technischen Merkmale werden unter anderem durch eine *Lebenszyklusgrafik* repräsentiert. In dieser werden die geplanten Versionsstände von IT-Werkzeugen auf eine Zeitachse aufgetragen und die aktuelle Lebenszyklusphase des Werkzeugs entsprechend dargestellt. Die *Schnittstellenarchitektur* beschreibt die Kommunikation zwischen verschiedenen IT-Werkzeugen. Die organisatorischen Merkmale werden durch die *Zuordnungstabelle* abgedeckt, welche Abhängigkeiten von zwei Betrachtungselementen in einer zwei-dimensionalen Matrix darstellt. Insbesondere kann hier die Systemverantwortung einer Organisationseinheit zugeordnet werden. Die prozessrelevanten Merkmale werden unter anderem durch die *Bebauungsplangrafik* repäsentiert, welche als besonders relevant gilt. Sie ist ebenfalls in Form einer Matrix aufgebaut und ordnet die IT-Werkzeuge entsprechend ihrer Nutzung den Organisationseinheiten zu. Somit ist es möglich, die Fähigkeiten eines IT-Werkzeugs zur Unterstützung eines Prozesse mit der tatsächlichen Nutzung dieser Fähigkeiten abzugleichen. Daneben beschreiben *Prozessdarstellungen* Abläufe von Geschäftsprozessen, z. B. mit Hilfe der Modellierungssprache OMG BPMN. Die Daten betreffenden Merkmale werden als Informationsobjekte nach IT-Werkzeugen separiert in OMG UML Klassendiagrammen mittels einer *Datenarchitektur* dargestellt. Letztlich wird über die *Qualitätsmerkmale* die Aufbereitung der Daten beschrieben, um Eingriffspunkte im Vorfeld der Integration zu kennnen [HH18, S. 994ff].

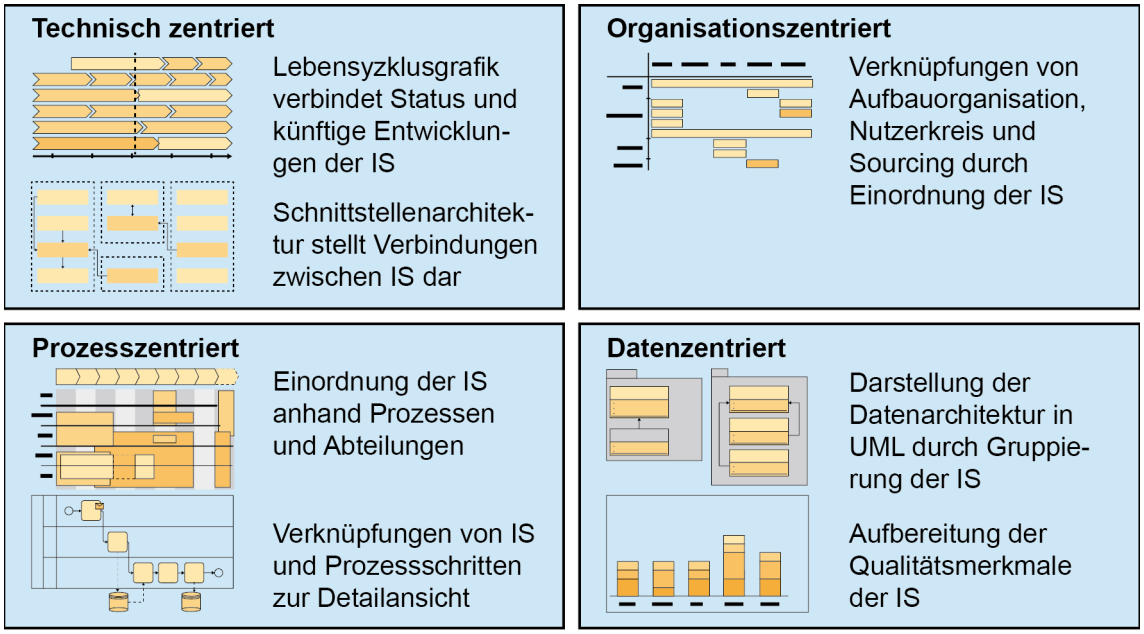


Bild 4-16: Übersicht über Sichten auf Informationssystemarchitekturen nach [HH18, S. 995]

Abschließend werden **Gestaltungsempfehlungen** für Informationssystem-Architekturen abgegeben, indem allgemeine Ziele postuliert werden. In die vier Dimensionen zusammengefasst sind dies [HH18, S. 997ff]:

- *Technische Merkmale:* Skalierbare Basisinfrastruktur (oft in der der Cloud); Datenaustausch zwischen IT-Werkzeugen über standardisierte Schnittstellen; Abgestimmtes und dokumentiertes Schnittstellenkonzept; Standort- und endgeräteunabhängiger Zugriff auf die Daten
- *Organisatorische Merkmale:* Architectural Thinking (Architekturentscheidungen gemeinsam zwischen IT-Abteilung und Fachbereich treffen); Bimodale IT-Abteilung (Aufteilung in traditionell gesteuerte IT für längerfristige Entwicklungen und agil gesteuerte IT für schnelle Anpassungen); Fokussierung kundenorientierter Prozesse
- *Prozessorientierte Merkmale:* Nutzung weniger, zentraler Informationssysteme; Anpassungen der zentralen Informationssysteme nur zur Wettbewerbsdifferenzierung; Bereitstellung schlanker Apps zur Anforderungserfüllung; Transparenz über Prozesse und Anwendungsfälle
- *Daten betreffende Merkmale:* Durchgängige Produktdatenstruktur auf Anwendungsebene; Zentrales Datenmodell; Redundanzfreiheit; Hohe Qualität der Informationen

Bewertung: Der Ansatz identifiziert grundlegende Potentiale und Variablen für die Gestaltung von Informationssystem-Architekturen in produzierenden Unternehmen. In Hinblick auf die Enterprise Architektur werden dabei alle vier Ebenen tangiert. Die vorgestellten Darstellungsformen sind typische Repräsentationen, die im Rahmen von EA-Modellen genutzt werden (vgl. [Han16, S. 58ff]) und hier für einen konkreten Anwendungsfall ausgewählt wurden. Dieser weist Schnittmengen mit der Gestaltung von Anwendungsarchitekturen für die Engineering-IT auf. Die Analyse von Anforderungen und die Evaluierung von Architekturen bzw. IT-Werkzeugen wird jedoch nicht betrachtet, weshalb die vorgestellten Darstellungsformen hierzu nicht zwangsläufig geeignet sind. Es wird darüber hinaus keine Methode oder Modellierungssprache bereitgestellt, die alle relevanten Aspekte abdeckt. Die Gestaltungsempfehlungen sind sehr abstrakt und allgemeingültig formuliert. Es ist nicht sichergestellt, dass die Gestaltungsempfehlungen für ein spezifisches Unternehmen sinnvoll oder überhaupt umsetzbar sind.

4.3.3 Modellbasierte Spezifikation einer MBSE-Modellierungsumgebung nach FRIEDLAND ET AL.

FRIEDLAND ET AL. beschreiben einen MBSE-basierten Ansatz zur Spezifikation von Modellierungswerkzeugen zur Umsetzung von MBSE. Der Ansatz entstand beim Flugzeughersteller Boeing, um Standardsoftware zu spezifizieren und umfangreich anzupassen. Die MBSE-Modellierungsumgebung, bestehend aus MBSE-Werkzeugen, wird dabei selbst als System betrachtet. Zur Spezifikation werden daher Grundlagen des SE genutzt und an die IT-Systementwicklung angepasst. Den Kern des Ansatzes bilden die Arbeitsschritte

um die in Bild 4-17 dargestellten Resultate zu erzeugen [FMH16, S. 1ff]. Der Ansatz beschreibt über diese Schritte hinausgehend ein unterstützendes Rollenmodell und einen Prozess zum Konfigurationsmanagement des Spezifikationsmodells [FMH16, S. 10ff].

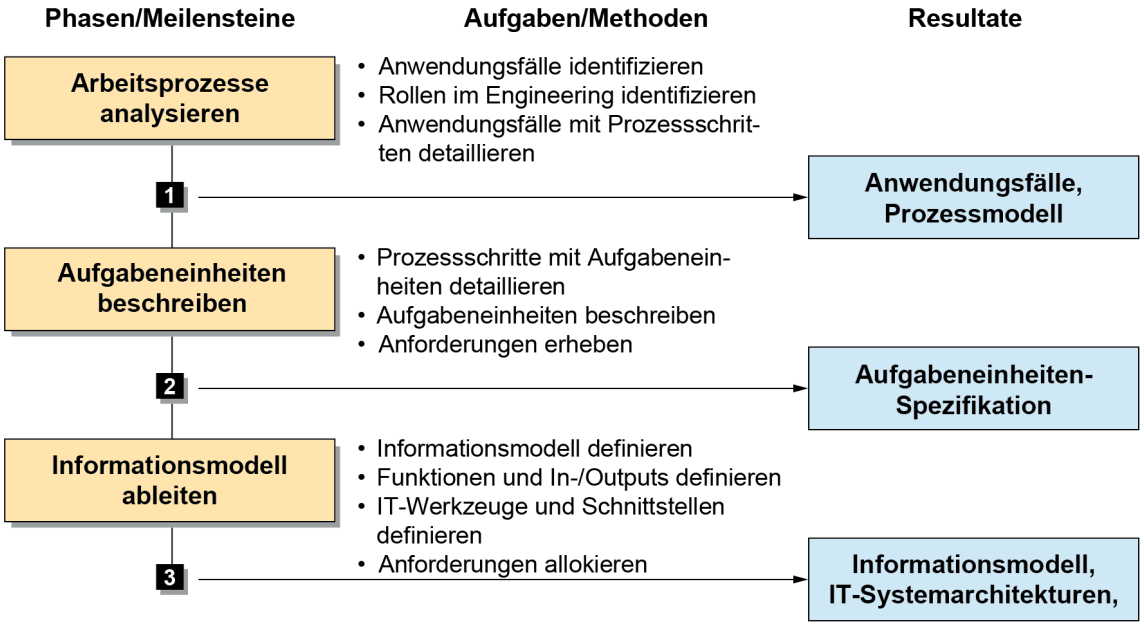


Bild 4-17: Vorgehensmodell zur SE-gestützten Spezifikation von MBSE-Modellierungswerkzeugen nach FRIEDLAND ET AL.

Zur Definition der **Arbeitsprozesse** (Work Processes) werden in einem SysML Use Case[1] Diagramm Anwendungsfälle aufgenommen, welche eine abstrakte Funktionalität innerhalb eines Geschäftsprozesse beschreiben, die durch das MBSE-Werkzeug unterstützt werden soll. Die den Rollen zugeordneten Anwendungsfälle werden in Prozessschritte untergliedert und mittels eines Prozessmodells visualisiert. **Aufgabeneinheiten** (Unit Tasks) beschreiben die Prozessschritte und ihre Flüsse im Detail. Zur Beschreibung der Aufgabeneinheit werden die Vor- und Nachbedingungen für das MBSE-Werkzeug sowie erwartete Ergebnisse und Ausnahmen bei nicht erfüllten Vorbedingungen definiert. Weiterhin werden funktionale Prozess- und Modellanforderungen (Wie wird das Modell bei Ausführung der Aufgabeneinheit manipuliert, welche Einschränkungen und Regeln soll es geben?), der geforderte Automatisierungsgrad sowie Qualitätsanforderungen (z. B. Gebrauchstauglichkeit) definiert [FMH16, S. 3f].

Basierend auf dieser Analyse wird ein **Informationsmodell** erstellt, welches als UML Klassendiagramm dargestellt wird. In diesem werden neben den Objekten und Sub-Objekten ihre Attribute und möglichen Beziehungen definiert. Zur Erfassung von Anforderungen an die Modellierung werden sogenannte Instanzdiagramme genutzt, die mit einer eigenen Notation ein abstraktes Abbild eines konkreten fachlichen Systemmodells darstellen. Ergänzend wird ein Konzept zur Ableitung von Architekturen vorgestellt. Dazu werden

[1] Der Begriff Use Case stammt aus dem Englischen und ist im Rahmen der Arbeit mit dem deutschen Begriff Anwendungsfall übersetzt.

Funktionen des MBSE-Werkzeugs definiert und die Anforderungen darauf allokiert. Die Funktionen werden über funktionale Schnittstellen mit In- und Outputs in Beziehung gesetzt. Nachfolgend werden implementierbare IT-Werkzeuge zu den Funktionen allokiert. Analog zu den Funktionen werden In- und Outputs sowie Werkzeugschnittstellen definiert. Die Werkzeugschnittstellen werden per Allokation mit den funktionalen Schnittstellen verknüpft [FMH16, S. 6ff].

Bewertung: Der Ansatz gibt Hilfestellung zur Analyse von Tätigkeiten und Prozessen im MBSE. Dazu werden geeignete Artefakte und ein Vorgehen beschrieben. Methoden des Engineerings werden jedoch nicht explizit berücksichtigt. Zwar wird dargestellt, wie Funktionen und IT-Werkzeuge auf die Analyseergebnisse allokiert werden können, ein methodischer Ansatz zur Unterstützung dieser Schritte ist jedoch nicht vorhanden. Die Evaluierung der Architektur und IT-Werkzeuge wird nicht betrachtet. Während Auszüge eines Metamodells dargestellt sind, bleibt ein übergeordnetes und definiertes Konzept zur Modellierung mit geeigneter Notation und Sichten offen.

4.3.4 Funktionale Architekturen für Systeme nach LAMM UND WEILKIENS

Die FAS-Methode (Funktionale Architekturen für Systeme) nach LAMM UND WEILKIENS ist ein Ansatz zur systematischen Ableitung von Architekturen für Systeme. Sie wird aktuell von der FAS-Arbeitsgruppe der GfSE[2] weiterentwickelt. Obgleich die Autoren als Anwendungsgebiet von FAS allgemein von Systemen sprechen, werden in den Anwendungsbeispielen ausschließlich technische Systeme genannt. Die Methode ist sprachenunabhängig, wird aber von den Autoren in Verbindung mit der SysML und einem eigenen Sprachprofil genutzt [LW14, S. 225f]. Die Anwendung von FAS kann in drei Schritte gegliedert werden, welche in Bild 4-18 dargestellt sind [LW10, S. 109ff].

Vorbereitend werden die Systemgrenze und externe Schnittstellen definiert sowie grundsätzliche Annahmen getroffen. Darauf basierend werden funktionale und nicht-funktionale **Anforderungen** an das System erhoben. Für die Anforderungserhebung wird auf existierende Methoden verwiesen. Im folgenden Schritt werden **Anwendungsfälle** identifiziert und durch Vor- und Nachbedingungen sowie die Auslöser und Resultate spezifiziert. Sie beschreiben lösungsneutral die zielorientierte Nutzung vom System bereitgestellter Dienste durch externe Akteure oder Systeme. Die Anwendungsfälle werden durch einzelne Aktivitäten definiert und Flüsse (Information, Signale, Material, Kraft, Energie) zwischen den Anwendungsfällen bzw. Aktivitäten modelliert [LW10, S. 111f], [LW14, S. 227f].

Während die vorangegangenen Schritte der Systemanalyse dienen, folgen Schritte zur Gestaltung der Systemarchitektur. Dazu werden die Anwendungsfälle in **funktionale Gruppen** überführt. Diese beschreiben eine Menge von stark zusammenhängender Aktivitäten. Zur Visualisierung können Funktionsbäume genutzt werden [LW14, S. 226ff],

[2] Gesellschaft für Systems Engineering, deutsches Chapter der INCOSE

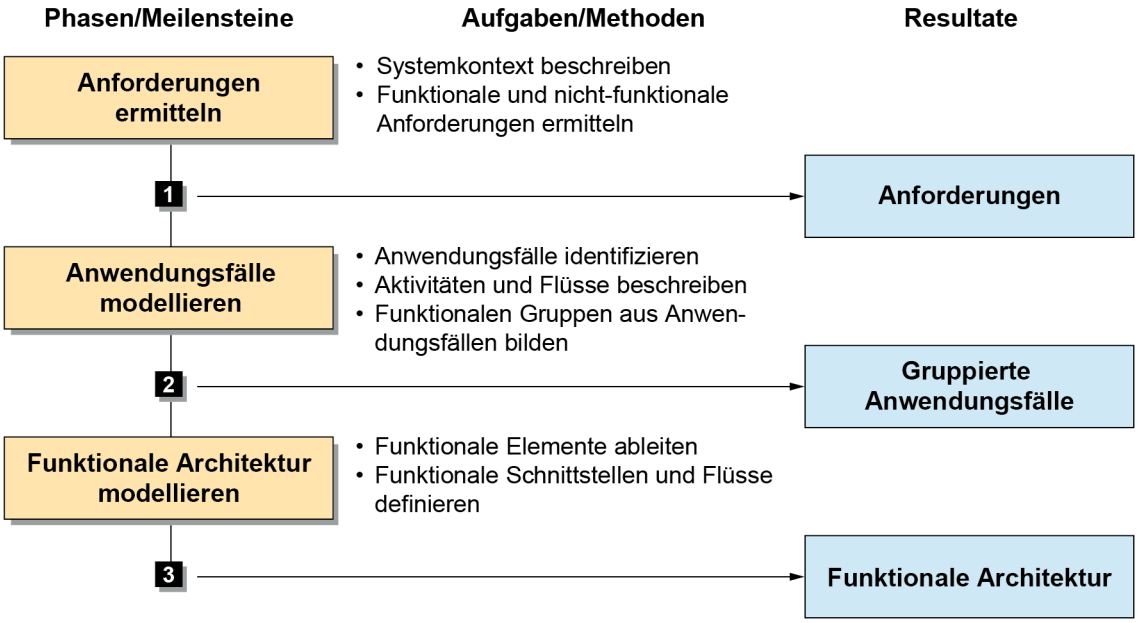


Bild 4-18: FAS-Vorgehensmodell zur Ableitung funktionaler Architekturen

[KLW12, S. 4]. Da die Ableitung nicht systematisch oder automatisiert durchgeführt werden kann, stellt FAS folgende Heuristiken bereit [LW14, S. 228f], [LW10, S. 113ff]:

- *Abstrakte und sekundäre Anwendungsfälle zusammenfassen:* Abstrakte Anwendungsfälle fassen Gemeinsamkeiten von konkreteren Anwendungsfällen zusammen. Sekundäre Anwendungsfälle beschreiben ein Systemverhalten, welches durch mehrere Anwendungsfälle genutzt wird.
- *Funktionen zur Aktor-Interaktion zusammenfassen:* Es kann sich anbieten, Funktionen zur Bereitstellung externer Schnittstellen in einer funktionalen Gruppe zusammenzufassen.
- *Sich gegenseitig aufrufende Funktionen zusammenfassen:* Ruft eine Funktion eine andere Funktion auf, geschieht dies meist in einem verwandten Kontext. Dadurch entstehen Netzwerke, aus denen funktionale Gruppen abgeleitet werden können.
- *Funktionen mit gleicher Datengrundlage zusammenfassen:* Bei der Nutzung spezifischer Daten (Artefakte) durch mehrere Funktionen wird davon ausgegangen, dass ein starker Zusammenhang zwischen diesen existiert. Aufschluss dazu geben bspw. die Flüsse zwischen Aktivitäten.
- *Bestehende Strukturen nutzen:* Da Systeme häufig evolutionär weiterentwickelt werden, kann hier ggf. auf die bestehenden Strukturen zurückgegriffen werden. Dabei muss jedoch sichergestellt sein, dass diese Struktur lösungsneutral und grundsätzlich sinnvoll ist.

Die funktionalen Gruppen werden anschließend in gleichnamige **funktionale Elemente** überführt. Sie sind losgelöst von den Anwendungsfällen und stellen die Elemente der

funktionalen Architektur dar. Diese entsteht, wenn die funktionalen Elemente über Flussbeziehungen miteinander verbunden werden. Funktionale Elemente können zudem in einer Baumstruktur gegliedert werden [LW14, S. 228], [KLW12, S. 5].

Bewertung: FAS bietet ein strukturiertes Vorgehen und Hilfsmittel wie Heuristiken zur systematischen Ableitung funktionaler Architekturen. Der Ansatz ist jedoch eher auf die Entwicklung von technischen Systemen ausgelegt und daher nur partiell geeignet, die Analyse der Engineering-IT zu unterstützen. Zur Erhebung von Anforderungen muss auf andere Ansätze zurückgegriffen werden. Da ausschließlich die funktionale Ebene betrachtet wird, bietet der Ansatz keine direkte Hilfestellung zur Ableitung physischer Architekturen und der Auswahl geeigneter implementierbarer Architekturkomponenten.

4.4 Ansätze zur Auswahl von IT-Werkzeugen

Das folgende Kapitel widmet sich Ansätzen zur Auswahl von IT-Werkzeugen bzw. IT-Architekturen. Die einschlägige Literatur führt zur präskriptiven Entscheidungsfindung, bei qualitativen, mehrdimensionalen Zielstellungen im betriebswirtschaftlichen Kontext häufig die Nutzwertanalyse nach ZANGEMEISTER oder der Analytic Hierarchy Process nach SAATY an [BCK19, S. 58ff], [BL12, S. 173f], [Mül19, S. 165ff], [Göt14, S. 193ff], [Ada96, S. 412ff]. Vereinzelt werden auch komplexere Verfahren wie Prävalenzverfahren [Göt14, S. 229ff], [BCK19, S. 61f], Multi-Attributive Nutzentheorie [Göt14, S. 217ff], [BL12, S. 173] oder Fuzzy Logik und neuronale Netze [Ada96, S. 421ff] genannt, die jedoch in praktischen Ansätzen zur Evaluierung von IT-Werkzeugen eine untergeordnete Bedeutung aufweisen. Im Folgenden werden Ansätze beschrieben, welche die Auswahl verschiedener Alternativen von Standardsoftware mit Hilfe einer Auswahl von Verfahren zur Entscheidungsfindung unterstützen.

4.4.1 Nutzwertbasierte Evaluationsansätze für Standardsoftware

3-Phasen-Konzept zur Auswahl von ERP-/PPS-Systemen nach SCHUH ET AL.

Für die Auswahl betriebswirtschaftlicher Standardsoftware existieren zahlreiche Ansätze. Sie dienen der Unterstützung bei einem vorliegenden Entscheidungsproblem, d.h. eine Festlegung auf eine von mehreren, sich ausschließenden Alternativen von IT-Werkzeugen. Ein Großteil der Ansätze basiert nach einer Analyse von WIESE [Wie98, S. 7] und eigenen Recherchen auf der Nutzwertanalyse, bspw. die Ansätze nach [SV11, S. 51ff], [Lan09, S. 379ff], [Lei15, S. 247ff], [SBM+12, S. 342ff] oder [HN12, S. 336ff]. Die Ansätze sind ähnlich aufgebaut, weshalb im Folgenden das in Bild 4-19 dargestellte 3-Phasen-Konzept nach SCHUH exemplarisch vorgestellt werden soll.

Zur Durchführung einer Systemauswahl muss im Rahmen der **Organisationsanalyse** ein Projekt eingerichtet werden. Dies umfasst die Definition der Projektaufgaben und -ziele und des Untersuchungsbereichs, die Bereitstellung entsprechender Ressourcen und Erstellung eines Projektplans [SBM+12, S. 313ff]. Die Prozess- und Strukturanalyse

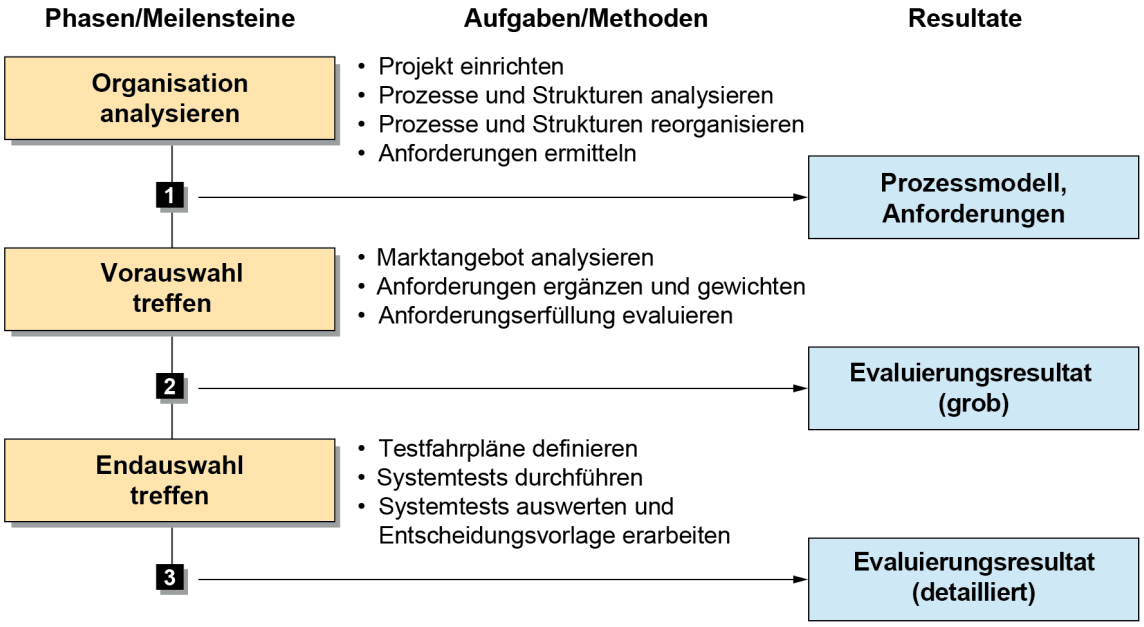


Bild 4-19: Das 3-Phasen-Konzept zur Auswahl von ERP-/PPS-Systemen nach SCHUH ET AL.

beinhaltet zunächst die Strukturierung der Prozesse im Untersuchungsbereich und die Erarbeitung eines quantitativen Datengerüsts, welches die Menge der später zu verwaltenden Daten beschreibt. Die verschiedenen Bestandteile der Struktur des Untersuchungsbereichs werden in einer Prozesslandkarte auf einem hohen Abstraktionsniveau zur Darstellung der Gesamtzusammenhänge mit ihren Informationsflüssen dokumentiert. Die einzelnen Teilprozesse der Prozesslandkarte können anschließend in Prozessmodellen detailliert werden. Dabei werden Schwachstellen identifiziert und zu Problemschwerpunkten zusammengefasst [SBM+12, S. 318ff]. Abschließend werden im Rahmen der Prozess- und Strukturreorganisation die Prozesse und Strukturen hinsichtlich der Optimierungspotentiale zu einem SOLL-Konzept angepasst. Darüber hinaus werden funktionale und systemtechnische Anforderungen an das IT-Werkzeug definiert. Abschließend ist die Implementierung des SOLL-Konzeptes zu planen [SBM+12, S. 324ff].

In der Phase der **Systemvorauswahl** wird zunächst das Marktangebot in Frage kommender IT-Werkzeuge hinsichtlich relevanter Eigenschaften untersucht. Die gesammelten leistungsbezogenen Anforderungen werden um strategische Anforderungen an das System und den Anbieter in einer Baumstruktur ergänzt. Alle Anforderungen werden gewichtet. Kritische Anforderungen fungieren als K.O.-Kriterien, da ein IT-Werkzeug bei Nichterfüllung auch nicht weiter evaluiert wird. Eine Menge ausgewählter IT-Werkzeuge und ihre Anbieter werden hinsichtlich ihrer Erfüllung der Anforderungen bewertet [SBM+12, S. 343ff].

Im Rahmen der **Endauswahl** werden sogenannte Testfahrpläne definiert. Diese beschreiben realitätsnahe Prozesse, mit denen eine möglichst große Menge an Funktionalitäten der IT-Werkzeuge geprüft werden kann. Dazu werden die Prozessmodelle aus der Organisationsanalyse genutzt und ein übergreifender Gesamtfahrplan und detailliertere Einzelfahrpläne erstellt. Die Testfahrpläne bestehen aus Aufgaben an den Toolvendor, deren

Lösungen im Rahmen einer Demonstration des IT-Werkzeugs mit Beispieldaten durchlaufen werden. Die Bestandteile des Testfahrplans werden durch die Teilnehmer bewertet und anschließend nachbereitet. Sie bilden die Grundlage für die Durchführung der Nutzwertanalyse. Die Ergebnisdarstellung sollte strukturiert und top-down (von Prozessen zu Funktionen) erfolgen, um einen Überblick über die Ergebnisse bekommen zu können. Darüber hinaus werden Referenzkunden besucht und alle Ergebnisse zur Erarbeitung einer Endauswahlentscheidung herangezogen [SBM+12, S. 349ff].

Allen eingangs genannten Ansätzen ist ähnlich, dass in einer frühen Phase Anforderungen bzw. Bewertungskriterien definiert und gewichtet werden. In den folgenden Phasen erfolgt ein meist zweistufiges Bewertungsverfahren. So beginnt bspw. das Vorgehen von SCHÜTTE UND VERING ebenfalls mit einer vorbereitenden Phase zur *Konstruktion des Auswahlproblems* und einer *Grobanalyse*, bei der Grob-Anforderungen mit den am Markt verfügbaren Systemen verglichen werden. In der letzten Phase zur *Detailanalyse* werden detaillierte Kriterien für die Nutzwertanalyse erhoben, zugehörige Bewertungsskalen festgelegt und die Kriterien hierarchisch strukturiert. Eine beispielhafte Nutzwertanalyse mit zu gewichteten Teilnutzenwerten aggregierten Kriterien ist in Tabelle 4-2 dargestellt. Für die Erhebung der Kriterien werden Techniken wie Interviews, Beobachtungen, Dokumentenanalysen oder Fragebögen vorgeschlagen. Neben der Nutzwertanalyse zur nicht-monetären Analyse ist eine monetäre Analyse mit Hilfe von Verfahren der traditionellen Investitionsrechnung vorgesehen [SV11, S. 65ff].

Tabelle 4-2: Beispielhafte Darstellung einer Nutzwertanalyse mit zu Teilnutzenwerten aggregierten Kriterien in Anlehnung an [SV11, S. 118]

Kriterien/ Anbieter	Anwendungs-bezogen (15%)	System-technisch (15%)	Anbieter-bezogen (15%)	Gesamt-nutzen
IT-Werkzeug 1	7	4	5	6,25
IT-Werkzeug 2	5	6	4	5
IT-Werkzeug 3	4	3	8	4,45

Bewertung: Die auf der Nutzwertanalyse basierenden Verfahren bieten ein strukturiertes Vorgehen. Durch die Aktivitäten zur Prozess- und Anforderungsanalyse wird ein grundsätzliches Verständnis der Problemstellung geschaffen. Die Nutzwertanalyse ist darüber hinaus einfach durchzuführen und unterstützt die Evaluation von IT-Werkzeugen. Es fällt jedoch auf, dass die Ansätze auf die Auswahl betriebswirtschaflich orientierter Standardsoftware ausgelegt sind. Keiner der genannten Ansätze bietet eine tiefergehende methodische Unterstützung um zu beschreiben, wie Anforderungen und Kriterien systematisch abgeleitet werden können. Obgleich in der Analyse häufig Prozessmodelle genutzt werden, fehlen übergreifende Modellierungskonzepte. Die Zusammenhänge mit der Architekturerstellung werden nahezu vollständig außer Acht gelassen. Dies führt dazu, dass Wechselwirkun-

gen zwischen Architekturgestaltung und Evaluierung kaum oder gar nicht berücksichtigt werden. Vielmehr sind die Ansätze darauf ausgelegt, eher monolithische gestaltete ERP-Systeme auszuwählen, die im Vergleich zu IT-Werkzeugen im Engineering weniger stark vernetzt sind.

Vorgehen zur Auswahl von MBSE-Werkzeugen nach FRIEDLAND ET AL.

FRIEDLAND ET AL. beschreiben einen bei Boeing entwickelten Ansatz zur Evaluierung von MBSE-Modellierungsumgebungen. Dies war notwendig, um eine extrem angepasste und veraltete MBSE-Modellierungsumgebung durch eine geeignete COTS Standardsoftware zu ersetzen. Der Ansatz weist keinen expliziten Zusammenhang mit dem in Kap. 4.3.3 beschriebenen Vorgehen auf. Er gliedert sich dazu in die drei in Bild 4-20 beschriebenen Phasen [FHF+17, S. 1f].

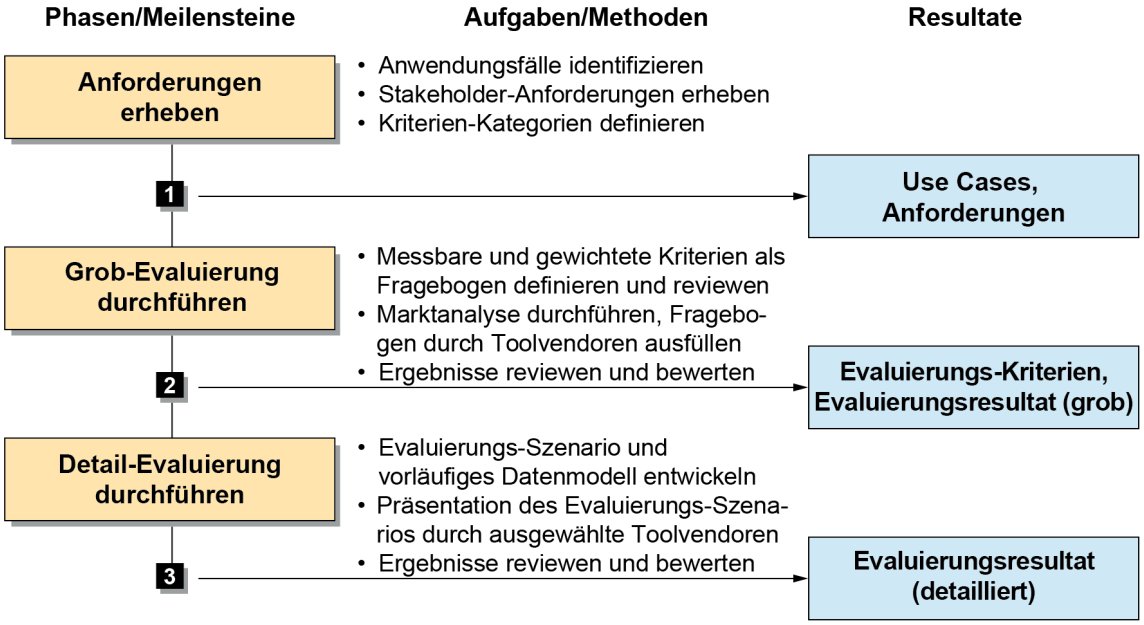


Bild 4-20: Vorgehensmodell zur Evaluierung von COTS MBSE-Modellierungswerkzeugen nach FRIEDLAND ET AL.

Zunächst werden in der Phase der **Anforderungserhebung** in mehreren Anwendungsfallkategorien die Hauptanwendungsfälle und diesen untergeordnete Anwendungsfälle identifiziert. Dazu wurden die Bedarfe der Hauptstakeholder über die Lebenszyklusphasen der MBSE-Modellierungsumgebung beobachtet. Zur weiteren Detaillierung werden Anforderungen aufgenommen und den Anwendungsfällen zugeordnet. Darauf basierend wurden Kategorien von Evaluierungskriterien abgeleitet [FHF+17, S. 3f].

In der folgenden Phase der **Grobevaluierung** werden aus den Anforderungen Evaluierungskriterien abgeleitet und in Form eines Fragebogens dokumentiert. Sie sind messbar, gewichtet, zu den Anforderungen rückverfolgbar und durch die Stakeholder gereviewt. Der so entstandene Fragebogen wird an bekannte Toolvendoren verschickt und durch diese hinsichtlich des Erfüllungsgrades ihres MBSE-Werkzeugs bewertet. Die Ergebnisse

werden im Evaluierungsteam gereviewt und etwaige Unklarheiten mit dem Toolvendor geklärt. Anhand der Resultate wird eine Vorauswahl getroffen, welche MBSE-Werkzeuge weiterhin in Frage kommen. Bei Nichterfüllung kritischer Kriterien (Showstopper) werden Werkzeuge von der weiteren Betrachtung ausgeschlossen [FHF+17, S. 5ff].

Die verbleibenden MBSE-Werkzeuge werden in der Phase der **Detailevaluierung** genauer untersucht. Dazu werden kritische Anwendungsfälle ausgewählt, auf deren Basis ein Evaluierungs-Szenario und ein vorläufiges Datenmodell erstellt und gereviewt werden. Anhand dieser vergleichbaren Bewertungsgrundlage demonstrieren die Toolvendoren die Umsetzung des Evaluierungs-Szenarios und der Anwendungsfälle mit ihrem MBSE-Werkzeug. Nach der Klärung offener Fragen wird jedes MBSE-Werkzeug durch das Evaluierungsteam und die Hauptstakeholder auf einer Ordinalskala von 0 bis 5 bewertet. Ausreißer in der Bewertung werden gemeinsam begutachtet und letztlich aufgrund der Gesamtbewertung Kombinationen ein oder mehrerer MBSE-Werkzeuge ausgewählt. Diese werden im Rahmen einer nachfolgenden Prototypen-Phase weiter getestet. Zwar konnte im Anwendungsbeispiel von Boeing die Anforderungserfüllung für das MBSE-Werkzeug maximiert werden, es waren jedoch weiterhin umfangreiche Anpassungen notwendig [FHF+17, S. 7ff].

Bewertung: Der Ansatz gibt ein klares Vorgehen zur Durchführung der Evaluierung vor. Dabei wird mehrstufig vorgegangen, um den Aufwand für die Evaluierung zu reduzieren. Der Ansatz gibt wenig methodische Unterstützung zur Ermittlung der Anwendungsfälle und Anforderungen. Die Ergebnisse im Rahmen des Ansatzes sind strukturiert, es wird jedoch nur teilweise beschrieben, wie diese dokumentiert werden. Ein übergreifender modellbasierter Ansatz wird nicht vorgestellt. Das Vorgehen beschreibt die Bewertung einzelner IT-Werkzeuge, die zusammenhängende und systematische Betrachtung im Kontext von IT-Architekturen wird jedoch nicht durchgeführt.

4.4.2 Alternative Evaluationsansätze für Standardsoftware

Die **VDI 2219** beschreibt einen Ansatz zur Einführung oder zum Wechsel von PDM-Systemen. Darüber hinaus beschreibt sie Grundlagen zu PDM-Systemen und ihrer betrieblichen Nutzung. Wesentlicher Bestandteil ist ein Vorgehensmodell zur Auswahl eines PDM-Systems [VDI2219, S. 2ff, 23].

Das Vorgehen umfasst fünf Schritte. Zunächst wird eine *Istanalyse* durchgeführt. Dabei werden die relevanten Prozesse, Daten- und Informationen, ihre Metadaten sowie genutzte Nummern- und Klassifizierungssystematiken analysiert. Diese Erkenntnisse fließen in das *Sollkonzept* ein, welches den Einsatz des PDM-Systems entlang der PLM-Prozesse ausgehend von den Unternehmenszielen beschreibt. Dazu werden ein Datenmodell festgelegt, die PDM-Funktionen einzelnen Prozessschritten zugeordnet und die künftige IT-Architektur hinsichtlich Infrastruktur und Datenverteilung beschrieben. Die *Anforderungsliste* wird aus dem Sollkonzept abgeleitet und dient der Auswahl eines PDM-Systems. Sie beschreibt die geforderten Funktionen zur Umsetzung des Sollkonzepts. In der *Systemauswahl* werden

drei bis vier PDM-Systeme detailliert untersucht. Dazu werden strategische und funktionale Kriterien sowie Referenzen und Anbieterqualität und -stabilität berücksichtigt. Anhand eines Drehbuchs, welches die gewünschte Arbeitsweise aus funktionaler Sicht anhand von Anwendungsfällen beschreibt, werden die Systeme vergleichbar bewertet. Dabei soll auch die Anpassungsfähigkeit über Parametrisierung und Customizing berücksichtigt und differenziert werden (vgl. Kap. 3.4.4). Die endgültige *Entscheidung* der Systemauswahl erfolgt im Rahmen einer Wirtschaftlichkeitsbetrachtung. Es werden dazu, in Abhängigkeit alternativer Bewertungsszenarien, geeignete Verfahren vorgestellt. Neben der monetären Bewertung, die im Rahmen dieser Arbeit nicht vertieft werden soll, wird zum Vergleich von PDM-Systemen neben der Nutzwertanalyse das Benefit Asset Pricing Model vorgeschlagen [VDI2219, S. 32].

Bewertung: Die VDI 2219 stellt ein grundsätzliches Vorgehensmodell für die Einführung von PDM-Systemen bereit. Diese sind zwar ein zentraler Baustein einer Engineering-IT-Architektur, das Zusammenspiel mehrerer stark vernetzter Werkzeuge wird jedoch nicht explizit betrachtet. Darüber hinaus stellt der Ansatz (abgesehen von Verweisen auf die Verfahren zur Nutzen- und Wirtschaftlichkeitsbewertung) keine unterstützende Methoden oder Modelle bereit.

Das **Benefit Asset Pricing Model** (BAPM) entstand im Rahmen der Dissertation von SCHABACKER an der Universität Magdeburg [Sch01]. Das Verfahren zielt auf die Bewertung von Nutzen aus Technologien oder IT-Werkzeugen ab und basiert auf Modellen für Kapitalmarktanlagen [VWZ+18, S. 602]. Das grundsätzliche Vorgehen erfolgt in vier Phasen [Sch01, S. 102]. Ein spezifisches Vorgehen zur Auswahl eines PDM-Systems ist in Bild 4-21 dargestellt.

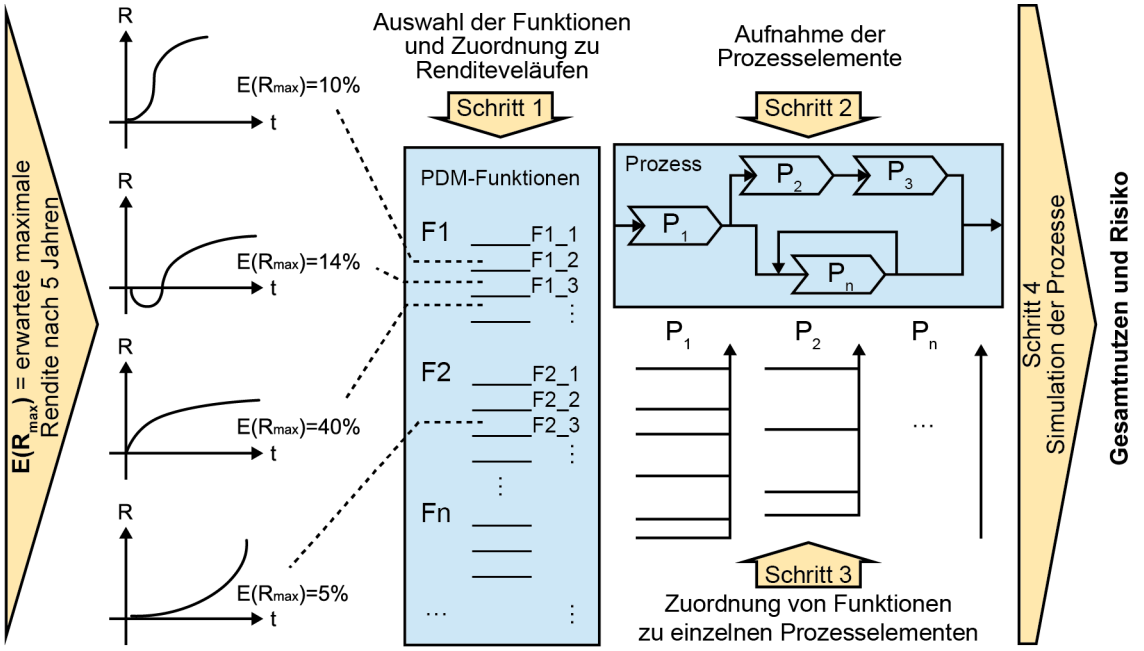


Bild 4-21: Vorgehensmodell zur Evaluierung PDM-Systemen mit dem BAPM-Verfahren in Anlehnung an [VWZ+18, S. 606], [SW02, S. 6]

In der *Nutzenklassenzuordnung* werden die Nutzen in gewichtete Nutzenklassen eingeordnet, welche direkt, direkt schwer, indirekt, indirekt schwer erfassbare Nutzen sowie Synergieeffekte differenzieren [VWZ+18, S. 606], [Sch01, S. 53ff]. Die Zuordnung ist elementar, da auf Basis der Nutzenklassen Analogien zu Kapitalmarktanlagen gezogen werden und auf dieser Basis entsprechende Rendite-Risiko-Profile zugeordnet werden. Diese greifen für quantifizierbare Nutzen auf finanzmathematische Verfahren zurück, bei nicht-quantifizierbaren Nutzen werden Erfahrungskurven herangezogen und der Nutzen durch Folgerungen mittels Expertenwissen quantifiziert [HHG13, S. 146], [Sch01, S. 55ff, 92ff]. Im Rahmen der PDM-Auswahl werden daher zunächst relevante Funktionen des Werkzeugs identifiziert und Renditeverläufen zugeordnet. Nachfolgend werden Prozesselemente aufgenommen. Die Prozesselemente werden den Werkzeugfunktionen zugeordnet. Im letzten Schritt der *Renditeberechnung* wird der zu erwartende Nutzen als Rendite berechnet. Im Beispiel der PDM-Auswahl wird der Prozess, bezogen auf einen Betrachtungszeitraum, simuliert und der Gesamtnutzen als Rendite errechnet und auf die Teilprozesse heruntergebrochen [VWZ+18, S. 606].

Bewertung: Das BAPM-Verfahren erlaubt die Berücksichtigung direkter und indirekter Nutzenbeiträge beim Vergleich von IT-Werkzeugen. Neben der Nutzenbewertung ist eine monetäre Betrachtung möglich, die im Rahmen dieser Arbeit jedoch nicht von Relevanz ist. Problematisch ist die hohe Komplexität der Methode in der praktischen Anwendung, was auch von mehreren Autoren angemerkt wird [Vol14, S. 26], [AG12a, S. 538]. Weiterhin müssen für die Erfassung nicht-quantitativer Nutzen entsprechende Erfahrungskurven vorliegen, deren Existenz oder Erstellung für den Kontext dieser Arbeit unklar bleibt. Dies ist insbesondere kritisch, da im Rahmen dieser Arbeit die nicht-monetäre Bewertung funktionaler und qualitativer Kriterien im Fokus steht.

4.5 Handlungsbedarf

Bild 4-22 zeigt die Bewertung der im Stand der Technik vorgestellten Ansätze hinsichtlich der in Abschnitt 3.6 definierten Anforderungen an eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*. Keiner der untersuchten Ansätze erfüllt die Anforderungen vollumfänglich. Der verbleibende Handlungsbedarf wird daher im Folgenden näher vorgestellt.

A1) Fokus auf das Engineering: Für die Anwendung im Kontext des Engineerings besteht die Anforderung, dass keine Anpassungen ohne Anleitung durchgeführt werden müssen. Diese Anwendung wird von einem Großteil der betrachteten Ansätze ganz oder teilweise erfüllt. Allerdings ist zu prüfen, dass bei der Nutzung von Ansätzen mit teilweiser oder nicht vorhandener Ausrichtung auf das Engineering eine entsprechende Anpassung an die Erfordernisse des Engineerings erfolgt.

A2) Erstellung der Prozess-, Methoden- und Datenarchitektur: Die methodisch angeleitete Erstellung einer Prozess-, Methoden- und Datenarchitektur als Anforderungsquelle ist essentiell. Keiner der untersuchten Ansätze erfüllt diese Anforderung vollumfänglich.

Die Ansätze nach HUTH und FRIEDLAND sowie TOGAF erlauben eine zusammenhängende Betrachtung von Prozessen und aus diesen resultierende Artefakte. Methoden werden jedoch nicht explizit betrachtet. Eine Integration bzw. Konsolidierung von verschiedenen Prozess- und Methodenbestandteilen wird somit nicht unterstützt. Ebenso ist die methodische Unterstützung zur systematischen Ableitung von Artefakten sowie ihrer Attribute und Beziehungen untereinander nur unzureichend gegeben. Explizite Hilfsmittel führt nur TOGAF auf, welche sich jedoch auf Prozesse und bereits bekannte oder genutzte Daten beschränkt. Es besteht somit Handlungsbedarf bei der Berücksichtigung von Methoden und der methodischen Ableitung der Datenarchitektur.

A3) Identifikation und Analyse von Anforderungen: Da eine Architektur basierend auf Anforderungen entwickelt und bewertet werden muss, soll die Identifikation und Analyse von Anforderungen für eine Engineering-IT-Architektur methodisch unterstützt werden. Hervorzuheben sind hier die Ansätze zum Requirements Engineering technischer Systeme, welche die Anforderungen in hohem Maße erfüllen und Techniken zur Identifikation, Dokumentation, Abstimmung und Verwaltung von Anforderungen bereitstellen. Allerdings beziehen sich diese auf die Entwicklung technischer Systeme, weshalb einige Techniken nicht anwendbar sind oder für den Kontext der Engineering-IT zu modifizieren sind. Die Ansätze nach HUTH und FRIEDLAND ermöglichen aufgrund einer zusammenhängenden Betrachtung die Ableitung von Anforderungen, die aus den Prozessen und Daten resultieren. Diese sind um eine Betrachtung der Methoden und Techniken zur systematischen Ableitung von Anforderungen zu erweitern.

A4) Ableiten geeigneter Applikationsarchitekturen: Die Ableitung geeigneter Applikationsarchitekturen soll systematisch und strukturiert unterstützt werden. Der Ansatz nach WEILKIENS erfüllt die Anforderung durch die Methode zur Ableitung funktionaler Architekturen anhand von Heuristiken. Diese sind allerdings auf die Entwicklung technischer Systeme und auf eine rein funtionale Betrachtung ausgelegt und somit für die Anwendung in der Engineering-IT zu modifizieren. CONSENS ist ebenfalls auf die Entwicklung von logischen Architekturen für technische Systeme ausgelegt, eine Abwandlung für IT-Systeme ist jedoch denkbar. Ansätze wie die von FRIEDLAND, HOFFMAN UND HEIMES sowie HUTH sehen zwar entsprechende Architekturen vor, machen aber keine näheren Angaben, wie diese methodisch hergeleitet werden können. Die semi-formale Spezifikation der Schnittstellen wird von keinem der betrachteten Ansätze abgedeckt.

A5) Anleitung zur Architektur-Evaluierung: Da typischerweise verschiedene Architekturalternativen denkbar sind, ist ein geeigneter Ansatz zur Evaluierung der Alternativen notwendig. Die untersuchten Ansätze betrachten jeweils nur einzelne IT-Werkzeuge und keine Kombinationen von IT-Werkzeugen im Rahmen einer Architektur. Sie sind daher auf eine differenzierte Bewertungsmöglichkeit von Applikationsarchitekturen anzupassen.

A6) Modellierungsrahmenwerk zur Spezifikation: Zur integrierten Modellierung aller Aspekte von der Anforderungsanalyse bis hin zur Architekturdefinition und -evaluation müssen eine Sprache und entsprechende Sichten vorliegen. Die betrachteten Ansätze

beschreiben Teilmengen oder Ausschnitte der darzustellenden Aspekte, z. B. ausschließlich Prozesse wie BPMN oder OMEGA. Je nach Sprache ist die Semantik eher abstrakt gehalten (UML/SysML, BPMN) oder konkret auf einen Anwendungsfall ausgeprägt (OMEGA, CONSENS, SPEM, ArchiMate, Spezifikationstechnik nach SCHNEIDER). ArchiMate deckt zwar den größten Umfang hinsichtlich der Semantik ab, insbesondere UML, SysML, SPEM und BPMN weisen jedoch leistungsfähige Erweiterungsmechanismen auf und lassen sich gut integrieren. Handlungsbedarf besteht daher hinsichtich der Erweiterung und ggf. Zusammenführung der Ansätze zu einer Sprache, die alle Aspekte integriert und eine einheitliche und intuitive Notation aufweist. Da keiner der Ansätze speziell auf die Entwicklung von Engineering-IT-Architekturen zugeschnitten ist, sind entsprechende Sichten zu definieren. Die ISO/IEC/IEEE 42010 stellt hierzu entsprechende Konzepte und Vorgaben bereit.

A7) Werkzeugunterstützung: Zur Bearbeitung des Modells ist eine Werkzeugunterstützung gefordert. Für die öffentlich spezifizierten Sprachen wie UML/SysML, SPEM sowie größtenteils auch ArchiMate steht für die produktive Nutzung geeignete Standardmodellierungssoftware mit entsprechenden Implementierungen bereit. Diese sind um ein Sprachprofil, geeignete Sichten und Digramme und, sofern möglich, entsprechende Unterstützung der Methode zu erweitern.

A8) Systematisches, methodisches Vorgehen: Ein systematisches Vorgehen soll den Anwender bei der Nutzung der Spezifikationstechnik anleiten. Diese Anforderung wird von einem Großteil der Ansätze vollständig oder in Teilen erfüllt. Lediglich einige Sprachen wie UML/SysML, BPMN, SPEM oder ArchiMate bieten kein Vorgehensmodell. Das Vorgehen ist im Hinblick auf die Integration der Ansätze und eventueller Wechselwirkungen unter diesen zu gestalten.

Keiner der betrachteten Ansätze und auch keine triviale Kombination einzelner Aspekte erfüllt alle der in Kapitel 3.6 definierten Anforderungen in vollem Umfang. Als entscheidende Schwachstelle kann die fehlende integrative und durchgängige Betrachtung aller Aspekte in einem Modell betrachtet werden. Dies geht mit mangelnder Unterstützung zur Ableitung der detaillierten Datenarchitekturen aus Prozessen und insbesondere den häufig nicht eingeschlossenen Methoden einher. Die konsistente Architekturgestaltung sowie -evaluierung werden nicht zusammenhängend betrachtet, sodass an dieser Stelle ebenfalls kaum methodische Unterstützung bereitsteht und Wechselwirkungen nicht berücksichtigt werden. Es besteht daher dahingehend Bedarf für eine *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen*.

Bewertung der untersuchten Ansätze hinsichtlich der gestellten Anforderungen.

Fragestellung:
Wie gut erfüllen die untersuchten Ansätze (Zeile) die gestellten Anforderungen an die Spezifikationstechnik (Spalte)?

Bewertungsskala:

○ = nicht erfüllt
◐ = teilweise erfüllt
● = voll erfüllt

		Anforderungen (A)							
		Fokus auf das Engineering	Analyse der Prozess-, Methoden- und Datenarchitektur	Identifikation und Analyse von Anforderungen	Ableiten geeigneter Applikationsarchitekturen	Anleitung zur Architektur-Evaluierung	Modellierungs-Rahmenwerk zur Spezifikation	Werkzeugunterstützung	Vorgehensmodell
		A1	A2	A3	A4	A5	A6	A7	A8
Ganzheitliche Ansätze zur Architekturbeschreibung	Beschreibung von Architekturen nach ISO/IEC/IEEE 42010	●	○	○	○	○	◐	○	◐
	Unified (UML) und Systems (SysML) Modeling Language	◐	◐	○	○	○	◐	●	○
	Spezifikationstechnik CONSENS	◐	○	◐	◐	○	◐	◐	●
	ArchiMate Modellierungssprache für Enterprise Architecture	○	◐	◐	○	○	◐	●	○
	Objektorient. Methode zur Geschäftsprozessmodellierung und -analyse (OMEGA)	◐	◐	◐	○	○	◐	◐	◐
	Business Process Model and Notation (BPMN)	○	◐	○	○	○	◐	●	○
	Software & Systems Process Engineering Metamodel (SPEM)	●	◐	○	○	○	◐	●	○
	Spezifikation von Wertschöpfungssystemen nach SCHNEIDER	◐	◐	◐	○	○	◐	◐	●
Analyse von Anforderungen und Prozessen	Ansätze zum Requirements Engineering Technischer Systeme	●	○	●	○	○	◐	◐	●
	Geschäftsprozessmanagement nach SCHMELZER UND SESSELMANN	○	◐	◐	○	○	○	○	●
	Model-Based Process Engineering nach HUTH et al.	●	◐	●	◐	○	◐	◐	●
Ansätze zur Definition von Enterprise und IT-Architekturen	The Open Group Architecture Framework (TOGAF)	○	◐	◐	◐	○	◐	◐	●
	Informationssystem-Architekturen nach HOFFMANN UND HEIMES	●	○	◐	◐	○	◐	○	○
	Modellbasierte Spezifikation einer MBSE-Umgebung nach FRIEDLAND et al.	●	◐	●	◐	○	◐	◐	◐
	Funktionale Architekturen für Systeme nach LAMM UND WEILKIENS	◐	○	○	●	○	◐	◐	●
Ansätze zur Auswahl von IT-Werkzeugen	3-Phasen-Konzept nach SCHUH (*Nutzwertbasiert*)	◐	○	○	○	◐	○	○	●
	Auswahl von MBSE-Werkzeugen nach FRIEDLAND et al. (*Nutzwertbasiert*)	●	○	◐	○	◐	◐	○	●
	VDI 2219 - Einführung und Betrieb von PDM-Systemen (*BAPM-basiert*)	●	○	◐	○	◐	○	◐	●

Bild 4-22: Bewertung der untersuchten Ansätze hinsichtlich der Anforderungen an die Spezifikationstechnik

5 Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen

Im Folgenden wird die *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen* beschrieben. Die Spezifikationstechnik hat den Anspruch, dem in der Problemanalyse ermittelten Bedarf und den Anforderungen (vgl. Kap. 3.5 und 3.6) gerecht zu werden sowie den im Stand der Technik identifizierten Handlungsbedarf (vgl. Kap. 4.5) zu erschließen. Die vorgestellte Spezifikationstechnik soll einen Beitrag zur Analyse der Engineering Prozesse und Methoden sowie der Gestaltung und Bewertung geeigneter Applikationsarchitekturen leisten. Dazu greift sie auf den Ansatz des Systems Engineering zurück (vgl. Kap. 3.4.7). Als Engineering-IT-Architektur wird im Sinne dieser Arbeit die Nutzung einer Enterprise Architektur (vgl. Kap. 3.4.8), bezogen auf die Prozesse, Methoden, Artefakte und IT-Werkzeuge im Engineering, verstanden. Sie verfolgt in diesem Zuge folgende Ziele:

- **Planung von Engineering-IT-Architekturen**, d.h. die zusammenhängende Beschreibung von Prozessen, Methoden, Artefakten sowie der Applikationsarchitektur und zugehörigen Beziehungen der Modellelemente. Ziel ist ein ganzheitliches Bild zur Analyse, Gestaltung und Evaluierung im Rahmen der Konzeptentwicklung.
- **Interdisziplinäres Kooperations- und Kommunikationsmittel** für alle beteiligten Akteure. Dies erfordert eine eindeutige, semi-formal definierte Sprache, die gleichzeitig intuitiv für verschiedene Disziplinen verständlich ist.
- **Systematisches Vorgehen**, d.h. die beteiligten Akteure werden angeleitet, die Modellierungsssprache effizient und effektiv für die Erstellung eines Modells anzuwenden. Für den Anwendungsfall zur Entwicklung einer Engineering-IT-Architektur stehen geeignete methodische Hilfsmittel zur Verfügung.

Zunächst wird in Kapitel 5.1 ein Überblick über die verschiedenen Bestandteile der entwickelten Spezifikationstechnik gegeben. Die darauffolgenden Kapitel erläutern diese detailliert. Das Vorgehensmodell in Kapitel 5.2 beschreibt die notwendigen Phasen, Tätigkeiten und Hilfsmittel, um eine Engineering-IT-Architektur zu modellieren. Zur Erstellung der Modelle wird das in Kapitel 5.3 vorgestellte Sprachkonzept herangezogen. Ein Metamodell definiert die zur Unterstützung des Vorgehens notwendigen Sprachelemente, zudem werden grundlegende Konzepte zur systematischen Bildung von Sichten und der sichtenübergreifenden Vernetzung des Modells erläutert. Die Sprachkonstrukte und ihre Anwendung werden detailliert beschrieben. Abschließend werden in Kapitel 5.4 zur Umsetzung des Sprachkonzepts ein papierbasiertes Kartenset sowie eine Softwarelösung zur rechnergestützten Modellierung auf Basis von UML, SysML, BPMN und SPEM vorgestellt.

5.1 Die Spezifikationstechnik im Überblick

Die im Rahmen dieser Arbeit entwickelte Spezifikationstechnik umfasst drei eng verzahnte Bestandteile: Ein Vorgehensmodell, ein Sprachkonzept und eine Werkzeugunterstützung (vgl. Bild 5-1).

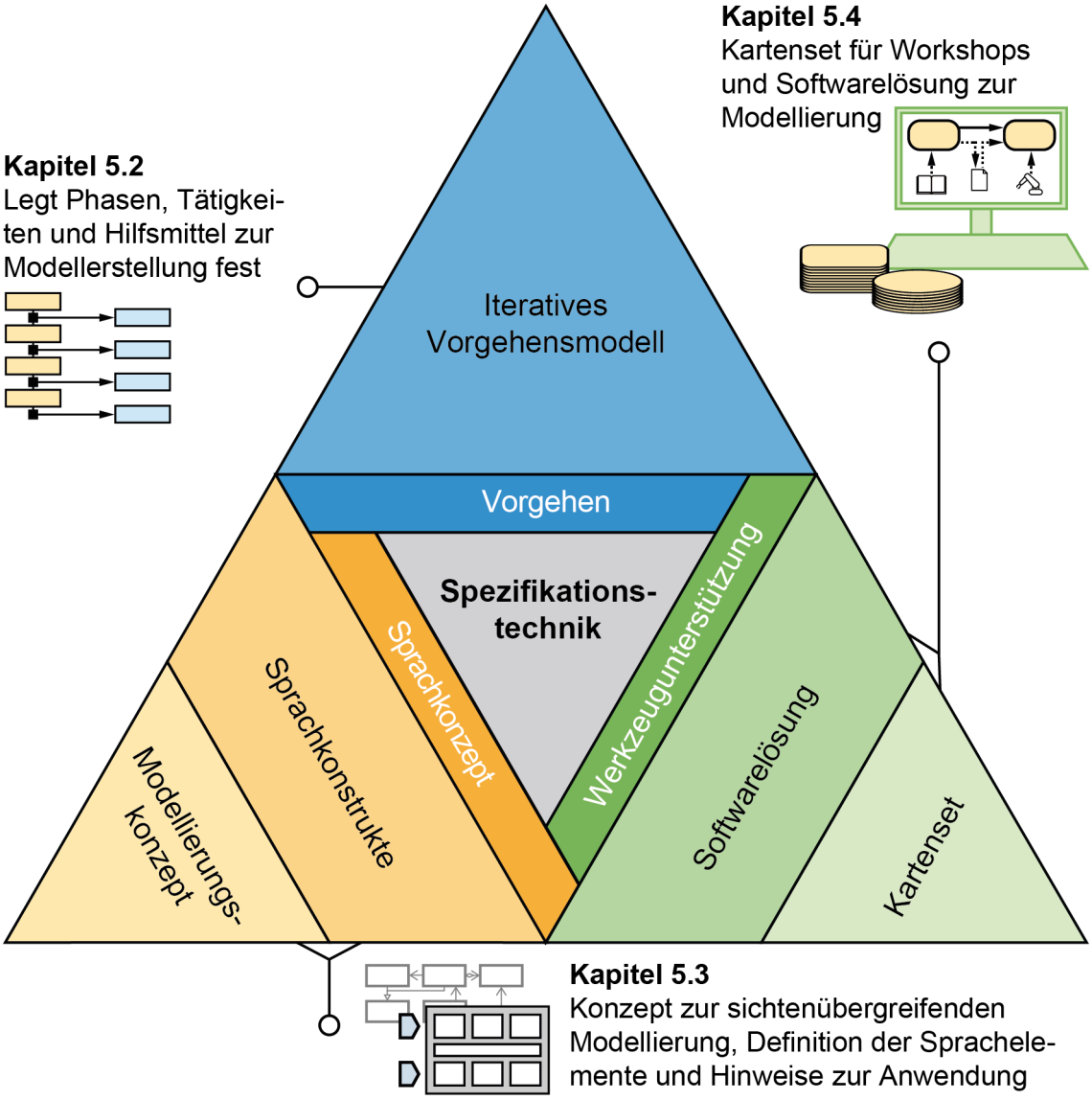


Bild 5-1: Bestandteile und Gliederung der Spezifikationstechnik im Überblick

Das **Vorgehensmodell** befähigt die beteiligten Akteure zur zielgerichteten Modellierung einer Engineering-IT-Architektur unter Anwendung des Sprachkonzepts. Es definiert hierzu eine idealtypische Beschreibung von Phasen, untergeordneten Schritten und unterstützenden Hilfsmitteln. Die Phasen bauen inhaltlich und modelltechnisch aufeinander auf.

Das **Sprachkonzept** im Sinne eines Modellierungsrahmenwerks dient der ganzheitlichen Beschreibung von Prozessen, Methoden, Modellen und Applikationsarchitekturen im Engineering. Dazu werden grundlegende Konzepte zur Anwendung der Modellierungssprache

festgelegt. Darunter fallen die Trennung zwischen Modell und Sicht sowie die systematische Bildung von Sichten mittels definierter Sichtweisen über aggregierte Partialmodelle oder fokussierte Viewpoints. Darüber hinaus wird ein Konzept zur sichtenübergreifenden Vernetzung der Modellinhalte vorgestellt. Weiterhin definiert das Sprachkonzept über die Syntax und Semantik die Modellierungssprache. Dies erfolgt im Rahmen eines Metamodells sowie der textuellen Definition der Sprachkonstrukte und Beschreibung ihrer Anwendung im Rahmen der Partialmodelle.

Die **Werkzeugunterstützung** erlaubt die Anwendung des Sprachkonzepts im Rahmen der praktischen Modellierung. Dazu wird ein Kartenset zur Umsetzung des Sprachkonzepts in Workshops zur Informationsaufnahme bereitgestellt. Zur Nutzung der Vorteile des modellbasierten Arbeitens ist eine rechnerinterne Repräsentation des Modells notwendig. Dazu wird ein Sprachprofil vorgestellt, welches die UML, SysML, BPMN und das SPEM-Profil um spezifische Stereotypen für Engineering-IT-Architekturen erweitert. Dadurch werden existierende Quasi-Standards zur Modellierung von Prozessen, Methoden und Systemen berücksichtigt. Das Sprachprofil setzt das Sprachkonzept und die Sprachkonstrukte um, indem es die Modellierungssprache formal definiert. Darüber hinaus wurde das Sprachprofil prototypisch in Form eines Plug-ins in ein kommerziell verfügbares Modellierungswerkzeug implementiert, um die praktische Anwendung im Modellierungswerkzeug durch angepasste Modellierungsfunktionen zu unterstützen.

5.2 Vorgehensmodell zur Spezifikation der Engineering-IT

Das im Folgenden vorgestellte Vorgehensmodell dient der Modellierung von Engineering-IT-Architekturen. Originäres Ziel ist die angeleitete Modellierung einer ganzheitlichen Engineering-IT-Architektur, welche die Prozesse, Methoden, Modelle und IT-Werkzeuge zusammenhängend beschreibt. Das Vorgehensmodell unterstützt die Erhebung der relevanten Modellinhalte und die evolutionäre Entwicklung des Modells durch logisch aufeinander aufbauende Modellbestandteile. Es beschreibt Phasen und Schritte zur iterativen Entwicklung des Modells.

Das Vorgehensmodell besteht aus vier in Bild 5-2 dargestellten Phasen mit untergeordneten Schritten. In der ersten Phase werden die Systemgrenzen definiert, Rahmenbedingungen und Informationsquellen identifiziert und aus ihnen der aktuelle Zustand der Prozesse und Methoden erfasst. Die zweite Phase dient der Analyse und Vervollständigung dieser Daten. Dabei werden Potentiale in den Prozessen bzw. Methoden identifiziert, erschlossen und durch Anwendungsfälle angereichert, sodass ein Zielbild für die Engineering Applikationsarchitektur entsteht. Die beiden ersten Phasen bilden somit die zur Ermittlung einer Lösung notwendige *Analyse*. Die dritte Phase verfolgt das Ziel, ein lösungsneutrales Konzept als Grundlage für die Lösungsfindung zu erarbeiten, welches in Kombination mit den Anforderungen als Zielsystem für die folgende Bewertung dient. Die Anforderungen sollten über alle Phasen hinweg identifiziert und dokumentiert werden. In der abschließenden vierten Phase werden konkrete, konsistente Lösungsvarianten definiert, in

eine Komponentenarchitektur überführt und hinsichtlich ihrer Zielerfüllung bewertet. Die beiden letzten Phasen umfassen somit die Gestaltung der Lösung, die *Synthese*. Die vier Phasen sollen im Folgenden näher vorgestellt werden.

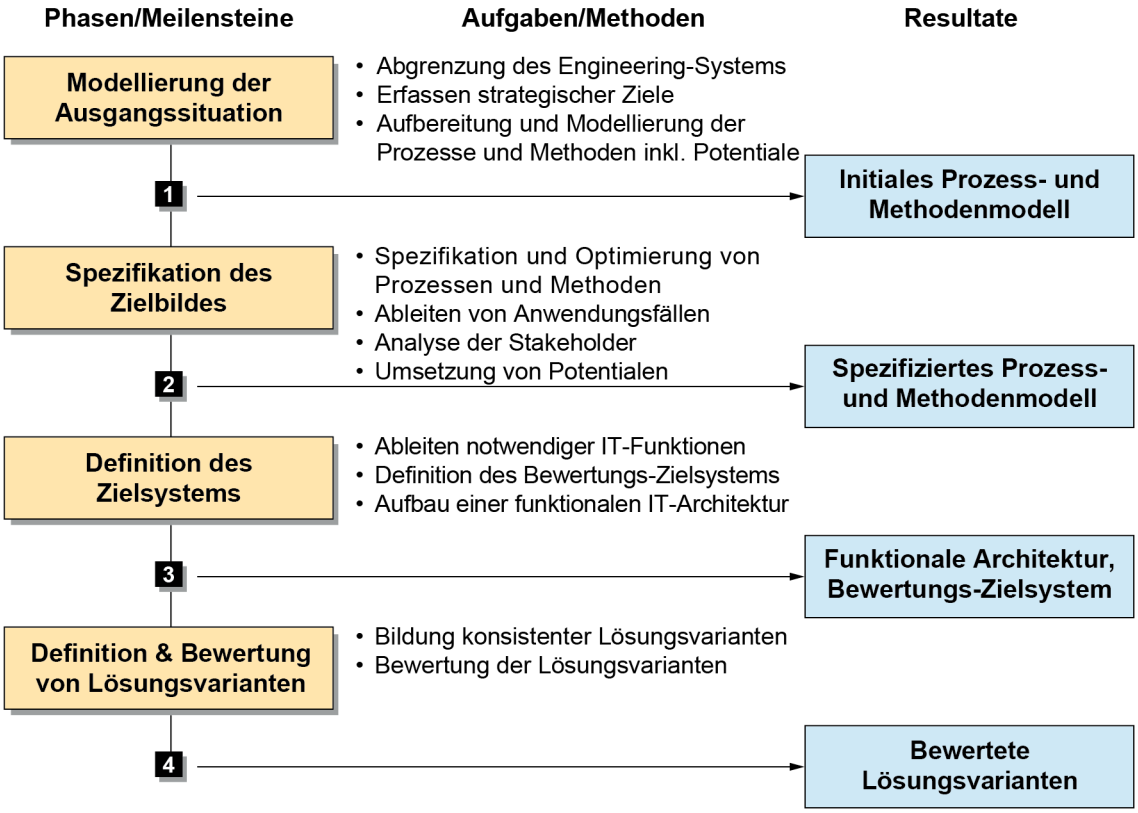


Bild 5-2: Das Vorgehensmodell im Überblick

5.2.1 Phase 1: Modellierung der Ausgangssituation

Zunächst wird im ersten Schritt die Aufgabe, die zur Modellierung der Engineering-IT-Architektur führt, geklärt und geplant. Dafür ist es notwendig, die **Systemgrenze** des Engineerings zu definieren, welche den Betrachtungsgegenstand gegenüber dessen Kontext abgrenzt. Die Modellierung erfolgt im Rahmen eines *Kontextmodells* (vgl. Kap. 5.3.2.2). Dazu werden zunächst die Organisationsbereiche, die Beziehungen zum Engineering aufweisen, identifiziert. Dies kann mit Prozessexperten des Engineerings erfolgen, es sollte jedoch auch mit in Frage kommenden Organisationsbereichen zur Gegenprüfung Rücksprache gehalten werden, ob Schnittstellen übersehen wurden oder zukünftig geplant sind. Anschließend werden die Beziehungen durch die Spezifikation der über Flüsse ausgetauschten Artefakte detailliert spezifiziert. Dies sollte in Zusammenarbeit mit den betroffenen Organisationsbereichen erfolgen, um eine konsolidierte Spezifikation der ausgetauschten Artefakte zu erhalten.

Der Entschluss, eine Engineering-IT-Architektur zu modellieren, gründet auf einer konkreten Motivation mit einhergehendem Bedarf. Oftmals steht diese Motivation in Zusammenhang mit strategischen Initiativen bzw. Projekten zur Umsetzung konkreter Ziele

(vgl. 4-Ebenen-Modell in Kap. 3.3.6). Diese **strategischen Ziele** sind daher zunächst zu identifizieren und im Modell zu erfassen. Sie können in allen Partialmodellen genutzt und mit deren Modellelementen verknüpft werden. Dies erlaubt es, bei Entscheidungen die Rückverfolgbarkeit zu den Zielen sicherzustellen.

Nachfolgend sind die innerhalb der Systemgrenzen des Engineerings liegenden und von den übergeordneten Zielen betroffenen relevanten **Prozesse** zu identifizieren. Die Prozesse werden strukturiert erfasst, indem die Aktivitäten und Artefakte im Rahmen eines *Prozessmodells* (vgl. Kap. 5.3.2.2) modelliert werden. Das Modell beschreibt den aktuellen Stand und berücksichtigt, falls bereits bekannt, die kurz- bis mittelfristig geplante Entwicklung von Prozessanpassungen[1]. Das Modell wird schrittweise detailliert, indem zunächst die Artefakte spezifiziert werden. Zudem werden die für die Ausführung der Aktivitäten herangezogenen Methoden identifiziert und modelliert. Im Kontextmodell identifizierte In- und Outputs für das Engineering werden an die jeweiligen Prozessaktivitäten weitergeführt. Dadurch wird sichergestellt, dass alle Schnittstellen später auch bedient werden können und im Prozess berücksichtigt werden. Zudem sollte identifiziert und modelliert werden, welche Lebenszyklusmanagement-Prozesse existieren und für welche Artefakte sie gültig sind.

Jede im Engineering genutzte **Methode** wird daraufhin separat erfasst und im *Methodenmodell* (vgl. Kap. 5.3.2.3) beschrieben. Dazu bieten sich Interviews mit den Methodenspezialisten und -anwendern an. Diese Technik ist grundsätzlich auch anwendbar, wenn keinerlei explizite Dokumentation zu den Methoden vorliegt. Falls doch, kann die Dokumentenanalyse der Methodenbeschreibungen wertvolle Informationen liefern. In diesem Rahmen werden die Methoden mit ihren Tätigkeiten, erzeugten Artefakten und den zwischen ihnen existierenden Beziehungen beschrieben.

Im letzten Schritt wird eine **Potentialidentifikation** auf Basis der bestehenden Modelle durchgeführt. Dies kann bspw. im Rahmen eines Workshops mit Prozess- bzw. Methodenverantwortlichen und -anwendern durchgeführt werden. Sie können bestehende Schwachstellen hinsichtlich der Engineering-IT-Architektur direkt im jeweiligen Partialmodell verorten[2]. Anschließend werden alle Potentiale hinsichtlich ihrer Relevanz priorisiert und die umzusetzenden Potentiale ausgewählt. Zur systematischen Priorisierung kann das in Kap. A3.1 beschriebene Vorgehen genutzt werden. Ergänzend kann festgelegt werden, dass nur Potentiale überhalb eines definierbaren Prioritäts-Schwellwert umgesetzt werden.

Abschließend wird ein Review für die Prozesse und Methoden auf Vollständigkeit und Korrektheit durchgeführt. Dazu ist sowohl implizites Wissen als auch explizites Wissen (z. B. in Form existierender Artefaktlisten) heranzuziehen. Als **Resultat** der Phase liegen

---

[1] Es gilt die Prämisse, dass zu ändernde oder neue Prozesse bzw. Methoden bekannt und definiert sind, da sie sonst nicht im Modell abgebildet werden können.

[2] Bei grundsätzlichen Potentialen innerhalb von Prozessen kann der Einsatz des Geschäftsprozessmanagements hilfreich sein (vgl. Kap. 4.2.2).

Modelle der Prozesse und Methoden vor, welche diese weitestgehend losgelöst voneinander beschreiben. Daher eignen sich diese Modelle in Kombination mit den strategischen Zielen noch nicht als Spezifikation für die IT-Werkzeuge, da z. B. Redundanzen nicht auszuschließen sind. Die Modelle dienen jedoch für die folgende Phase als Grundlage zur Analyse hinsichtlich Redundanzen sowie Durchgängigkeit und notwendiger Werkzeugunterstützung.

5.2.2 Phase 2: Spezifikation des Zielbildes

Die Prozess- und Methodenmodelle werden in dieser Phase systematisch analysiert und detailliert. Dadurch soll erreicht werden, dass sie als Grundlage für eine konsistente Spezifikation für die IT-Werkzeuge genutzt werden können. Darüber hinaus werden die Prozesse und Methoden auf durch von IT-Werkzeugen zu unterstützende Tätigkeiten hin analysiert.

Im ersten Schritt erfolgt die **Prozessspezifikation** auf Basis des *Prozessmodells* (vgl. Kap. 5.3.2.2). Hier gilt es, die geforderte Durchgängigkeit der Artefakte zwischen den Prozessaktivitäten zu spezifizieren. Dies sollte durch die Verantwortlichen der für die miteinander in Relation stehenden Prozessaktivitäten erfolgen. Zudem sollte geprüft werden, ob die Lebenszyklusmanagement-Prozesse ggf. konsolidiert werden sollten und ob alle relevanten Artefakte durch einen Lebenszyklusmanagement-Prozess gesteuert werden.

Im folgenden Schritt wird eine **Methodenspezifikation** durchgeführt, wozu das *Methodenmodell* (vgl. Kap. 5.3.2.3) herangezogen wird. Dabei werden die Methoden nicht mehr isoliert betrachtet, sondern im Kontext mit anderen Methoden und Artefakten. Grundlage dazu bieten die Kurzbeschreibungen der jeweiligen Artefakte, die auf Schnittmengen bzw. Redundanz untersucht werden. Wird eine Redundanz festgestellt, wird diese entsprechend im Modell gekennzeichnet. Anschließend werden die Redundanzen behoben. Dazu wird für jede Redundanz entschieden, in welche Verantwortlichkeit das zukünftige Artefakt wechselt. Artefakte, die bislang Redundanzen aufweisen, referenzieren stattdessen zukünftig auf das Artefakt bzw. dessen granulare Bestandteile. Dieser Sachverhalt wird durch entsprechende Beziehungen zwischen dem Artefakt und dessen Repräsentation modelliert.

Nachfolgend wird eine **Anwendungsfallanalyse** durchgeführt. Ziel ist es, auf Basis der Prozesse und Methoden allgemeine, wiederkehrende Tätigkeiten zu ermitteln und zu dokumentieren. Diese sollen durch ein IT-Werkzeug unterstützt und darin durchlaufen werden. Die resultierenden Anwendungsfälle werden im Modell festgehalten (vgl. Kap. 5.3.2.4).

Für die **Stakeholderidentifikation** werden zunächst Rollen in den verschiedenen Partialmodellen identifiziert. Anschließend werden diese in den Partialmodellen mit den Tätigkeiten der Prozesse, Methoden und Anwendungsfälle verknüpft. Daraus ergibt sich ein konsistenter Satz von Rollen, für den anschließend konkrete Personen hinterlegt werden sollen. Dadurch können bei weiterem Informationsbedarf zu spezifischen Themen gezielt

Ansprechpartner identifiziert werden. Zudem können die Rollen als Grundlage für weiterführende Analysen genutzt werden, z. B. um festzustellen, wem bestimmte IT-Werkzeuge zugewiesen oder Berechtigungen erteilt werden müssen.

Im folgenden Schritt wird eine **Potentialumsetzung** der in der Phase 1 identifizierten Potentiale aus den Prozessen und Methoden in den bestehenden Modellen durchgeführt. Es gilt, für alle umzusetzenden Potentiale geeignete Lösungen zu finden und das Modell dementsprechend anzupassen.

Abschließend wird ein Review der Prozesse, Methoden, Anwendungsfälle und Stakeholder auf Vollständigkeit und Korrektheit durchgeführt. Ebenso wird überprüft, ob die Engineering-IT-Architektur Redundanzen enthält, z. B. weil zwei Rollen unbewusst in mehreren Methoden genutzt wurden und hierfür mehrere Modellelemente angelegt wurden. Zum Abschluss dieser Phase dürfen keine Konflikte mehr hinsichtlich der Spezifikation von Artefakten und der Durchgängigkeit vorliegen, für Potentiale sollten entsprechende Lösungen vorliegen. **Resultat** der Phase ist ein konsistentes Modell der Prozesse, Methoden, Anwendungsfälle und Rollen, welches den Zielzustand beschreibt, auf dessen Grundlage die Entwicklung der IT-Lösung erfolgt.

5.2.3 Phase 3: Definition des Zielsystems

Die bisher erarbeiteten Modelle dienen als Grundlage für die Entwicklung eines Zielsystems, an welchem die in Phase 4 entwickelte IT-Lösung bewertet werden kann. Darüber hinaus wird eine funktionale Beschreibung der IT-Lösung erarbeitet. Funktionale Beschreibungen erlauben eine umfassende Beschreibung eines Systems, obwohl lösungsspezifische Details noch nicht bekannt sind [WLR+15, S. 92]. Darüber hinaus werden die Anforderungen an die IT-Lösung ins Zielsystem eingeordnet.

In einem ersten Schritt, der **Funktionsidentifikation,** werden durch die IT-Architektur bereitzustellende Funktionen aus den bisher vorliegenden Modellen abgeleitet. Diese werden im Rahmen der *Funktionshierarchie* (vgl. Kap. 5.3.2.5) strukturiert und modelliert. Zur methodischen Ableitung der Funktionen sind ebenda entsprechende Heuristiken beschrieben. Daraus resultiert eine hierarchische Struktur von Funktionen.

Darauf aufbauend wird im nächsten Schritt die **Zielsystemdefinition** durchgeführt. Auf Basis der bisher erarbeiteten Modelle werden Anforderungen, welche auch als Bewertungskriterien fungieren, identifiziert und dokumentiert. Die Anforderungsermittlung folgt dem Prinzip des RE als phasenübergreifende Aktivität und kann somit in allen Phasen und Arbeitsschritten des Vorgehensmodells erfolgen (vgl. Kap. 4.2.1). Spätestens an diesem Punkt werden alle *Anforderungen* (vgl. Kap. 5.3.2.6) einer dazugehörigen Funktion zugeordnet. Daraus resultiert ein hierarchisches Zielsystem, bei dem die Anforderungen die atomaren Elemente darstellen. Analog zu den Anforderungen werden die *Anwendungsfälle* (vgl. Kap. 5.3.2.4) den jeweiligen Funktionen untergeordnet.

Im letzten Schritt erfolgt die **Funktionsarchitektur-Modellierung**, indem die Funktionen miteinander in Beziehung gesetzt werden. Dazu wird eine *funktionale Architektur* (vgl. Kap. 5.3.2.5) genutzt. Diese greift auf die Hierarchie der Funktionen zurück.

Abschließend wird ein Review der funktionalen Architektur und der Anforderungen hinsichtlich Vollständigkeit und Korrektheit durchgeführt. Die Anforderungen sind in einem Review im Sinne der Kernaktivität „Anforderungen prüfen und abstimmen“ des RE durchzuführen. Dazu ist die Einhaltung der Qualitätskriterien an Anforderungen sowie die Konfliktfreiheit sicherzustellen (vgl. Kap. 4.2.1). Als **Resultat** der Phase liegt eine funktionale Architektur inkl. Funktionshierarchie sowie das Zielsystem zur Bewertung von IT-Lösungen vor.

5.2.4 Phase 4: Definition und Bewertung von Lösungsvarianten

Die funktionale Architektur stellt die Grundlage zur Lösungsfindung dar. Auf ihrer Basis werden mit Hilfe einer Morphologie geeignete Softwarekomponenten zur Umsetzung der Funktionen identifiziert. Eine konsistente Kombination von Softwarekomponenten stellt eine Lösungsvariante dar. Die Lösungsvarianten werden durch eine Komponentenarchitektur beschrieben, welche die Softwarekomponenten miteinander in Beziehung setzt. Jede Lösungsvariante wird abschließend hinsichlich der Erreichung der Anforderungen bewertet. Damit folgt das Vorgehen grundlegenden Prinzipien des SE, insbesondere dem Denken in alternativen Lösungen [HWF+18, S. 53], [WRF+15, S. 47ff].

Bildung konsistenter Lösungsvarianten

Im ersten Schritt, der **Lösungsraumdefinition,** werden prinzipiell geeignete Softwarekomponenten (IT-Werkzeuge bzw. deren Bestandteile) zur Umsetzung der Funktionen identifiziert. Die Modellierung erfolgt im Rahmen des *Lösungsmodells* (vgl. Kap. 5.3.2.7). Dazu werden zunächst geeignete Arten von IT-Werkzeugen identifiziert. Für diese werden konkrete Softwarekomponenten identifiziert und den Arten zugeordnet. Im Rahmen einer Marktrecherche kann der Beitrag einer Softwarekomponente zu Funktionsgruppen und Funktionen zur informellen Vorbewertung abgeschätzt werden[3]. Bei Bedarf können Rückfragen an die Softwarevendoren gestellt oder Präsentationen durch diese durchgeführt werden. Durch die Zuordnung von Softwarekomponenten zu Softwaregruppen ensteht ein Lösungsraum, der als Liste potentieller Softwarekomponenten dokumentiert ist.

Im Schritt der **Lösungsvariantenbildung** wird im *Lösungsmodell* zunächst geplant, welche Lösungsvarianten erzeugt werden sollen. Eine Lösungsvariante ist eine konsistente Kombination von Softwarekomponenten, um möglichst alle Funktionen und Anforderungen zu erfüllen. Zunächst wird geplant, wie viele Lösungsvarianten erstellt werden sollen,

[3] Sie muss aus Gründen der Praktikabilität nicht zwangsweise im Modell abgelegt werden, dies ist aber bspw. über Kommentarfunktionen von Modellierungswerkzeugen möglich.

wie sich diese differenzieren und ob bestimmte Softwarekomponenten zu berücksichtigen sind. Dies ist wichtig, damit die Lösungsvarianten sinn- und planvoll gebildet werden und um die klare Differenzierung der Varianten in der Modellierung zu unterstützen. Dabei muss auch die Wirtschaftlichkeit berücksichtigt werden, da eine Evaluierung aller theoretisch möglichen Varianten sehr aufwändig ist. Anschließend erfolgt die Zuordnung der Softwarekomponenten auf die Funktionen, unterstützt durch einen morphologischen Kasten. Der Schritt kann mehrfach wiederholt werden, um verschiedene Lösungsvarianten zu definieren. Stellt sich heraus, dass eine Funktion durch mehrere Softwarekomponenten realisiert werden kann, so ist zu prüfen, ob diese Funktion aufgeteilt werden sollte.

Da zur Definition konsistenter Lösungsvarianten die Beurteilung der Interoperabilität zwischen den Softwarekomponenten zur Umsetzung der Flüsse zwischen den Funktionen entscheidend ist, erfolgt iterativ, aufbauend auf den Ergebnissen der Lösungsvariantenbildung, die **Komponentenarchitektur-Modellierung**. Diese setzt die Softwarekomponenten über Schnittstellen im Rahmen der *Komponentenarchitektur* (vgl. Kap. 5.3.2.8) miteinander in Beziehung. Darüber hinaus werden die zu nutzenden Schnittstellen spezifiziert.

Bewertung der Lösungsvarianten

Im nachfolgenden Schritt erfolgt die **Lösungsvariantenbewertung**. Hinsichtlich der Evaluierung einer Engineering-IT-Architektur stellt sich die Frage, nach welchen Kriterien bzw. Dimensionen eine Bewertung erfolgen soll. Bezüglich der Dimensionen der Bewertung lässt sich der internationale Standard ISO/IEC 25010 heranziehen [ISO25010]. Dieser definiert ein Modell zur Beschreibung der Qualität von Software- und IT-Systemen. Als Qualität wird dabei der „Grad, in dem das System die erklärten und angedeuteten Bedürfnisse seiner verschiedenen Interessengruppen erfüllt und somit einen Wert liefert“ bezeichnet [ISO25010, S. 2]. Die Norm definiert dazu ein Qualitätsmodell, welches sich wie in Bild 5-3 dargestellt untergliedert.

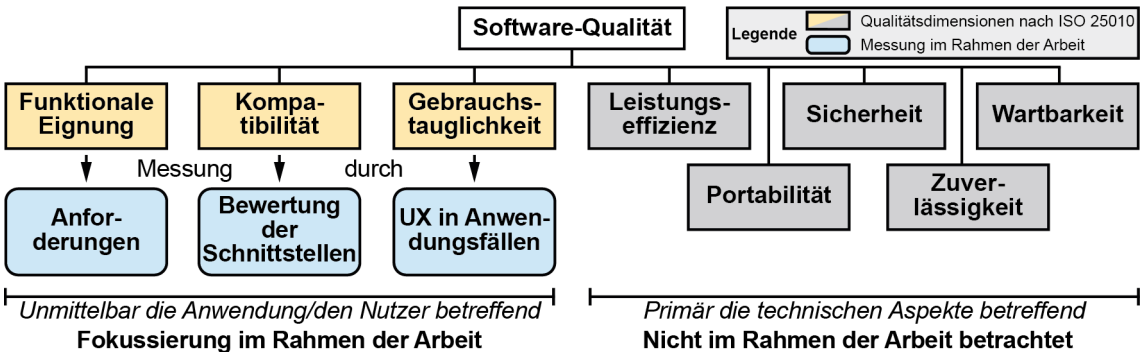


Bild 5-3: Modell zur Messung der Qualität von Softwaresystemen nach ISO/IEC 25010 und im Rahmen der vorliegenden Arbeit genutzte Messungen

Funktionale Eignung: Sie umfasst den „Grad, in dem ein Produkt oder System Funktionen bietet, die den angegebenen und impliziten Bedürfnissen entsprechen, wenn es unter definierten Bedingungen verwendet wird“ [ISO25010, S. 10f]. Daraus ergibt sich, dass

die Bedürfnisse unter den gegebenen Bedingungen definiert werden müssen («Angemessenheit» und «Korrektheit»). Im Rahmen der Spezifikationstechnik erfolgt dies über die Modellierung und Spezifikation der relevanten *Anforderungen*, die als Bewertungskriterien genutzt werden (vgl. Kap. 5.3.2.6). Sie umfassen funktionale Anforderungen und bei Bedarf Qualitätsanforderungen zur Detaillierung einer entsprechenden Funktion. Die Anforderungen werden gewichtet und anhand einer Skala bewertet, woraus Teilnutzenwerte für die Anforderungen und Funktionen resultieren. Zudem muss für ein spezifisches IT-Werkzeug die Abdeckung der Anwenderaufgaben über die festgelegten Funktionen und Anforderungen geprüft werden («Vollständigkeit»).

Kompatibilität: Sie beschreibt den „Grad, in dem ein Produkt, System oder eine Komponente Informationen mit anderen Produkten, Systemen oder Komponenten austauschen und/oder die erforderlichen Funktionen ausführen kann, während sie die gleiche Hardware- oder Softwareumgebung nutzt“ [ISO25010, S. 11f]. Dazu ist es entscheidend, in welchem Grad verschiedene IT-Werkzeuge Informationen austauschen und gemeinsam nutzen können («Interoperabilität»). Der erforderliche Grad der Kompatibilität wird daher systematisch in mehreren Partialmodellen abgeleitet und spezifiziert (vgl. Kap. 5.3.2.2, 5.3.2.3 und 5.3.2.5). Die Bewertung erfolgt anhand der Schnittstellen im Rahmen der *Komponentenarchitektur*, welche gewichtet und bewertet werden, woraus sich Teilnutzenwerte ergeben. Nach der ISO/IEC 25010 wird im Rahmen der Kompatibilität auch beurteilt, ob mehrere IT-Werkzeuge ihre Funktionen auch in einer gemeinsamen Umgebung mit geteilten Ressourcen bereitstellen können («Koexistenz»). Dieser Aspekt wird im Rahmen der Arbeit nicht berücksichtigt, da er im Regelfall durch Gestaltung der Infrastrukturarchitektur (z. B. mittels Virtualisierung) sichergestellt werden kann [LG20, S. 387ff].

Gebrauchstauglichkeit: Ziel der Engineering-IT ist primär, den Anwender bei seinen Aufgaben in der Anwendung von Prozessen und Methoden zu unterstützen, wozu eine entsprechende Gebrauchstauglichkeit der IT-Werkzeuge notwendig ist. Diese wird im Englischen auch als Usability bezeichnet und beschreibt den „Grad, in dem ein Produkt oder System von bestimmten Benutzern verwendet werden kann, um bestimmte Ziele mit Effektivität, Effizienz und Zufriedenheit in einem bestimmten Nutzungskontext zu erreichen“ [ISO25010, S. 12f]. Darunter fallen unter anderem Aspekte wie die «Erlernbarkeit», «Fehlervermeidung» oder «Bedienbarkeit». Die Gebrauchstauglichkeit kann durch das Konstrukt der User Experience (UX) erweitert werden. Diese ist definiert als „die Wahrnehmungen und Reaktionen des Benutzers, die aus der tatsächlichen und/oder der erwarteten Benutzung eines Systems, eines Produkts oder einer Dienstleistung resultieren“ [DIN9241-11, S. 22]. Während die Gebrauchstauglichkeit die wahrgenommene Nutzung während der Nutzung umfasst, erfasst UX auch die antizipierte Nutzung vor der Nutzung und die verarbeitete Nutzung nach der Nutzung [Sar11, S. 22]. Um verschiedene IT-Werkzeuge hinsichtlich der UX vergleichen zu können, soll im Rahmen dieser Arbeit auf den UEQ (User Experience Questionaire) zurückgegriffen werden. Dieser Ansatz nutzt einen standardisierten Fragebogen, bestehend aus 26 Items auf einer siebenstufigen Skala [Sar11, S. 187ff]. Dazu werden die nach Funktionen strukturierten Anwendungsfälle,

welche jeweils reproduzierbare Situationen beschreiben, gewichtet und anhand des UEQ bewertet. Die Nutzung des UEQ ist in Kap. A3.2 detaillierter beschrieben. Das Ergebnis der Bewertung wird in Teilnutzenwerte transformiert.

Die Bewertung der funktionalen Eignung und der Gebrauchstauglichkeit/UX sollte auf Ebene einzelner Funktionen erfolgen, um die IT-Werkzeuge granular bewerten zu können. Dies ermöglicht es, verschiedene Alternativen detailliert voneinander abgrenzen zu können. Gleichzeitig ist dadurch auch die Vergleichbarkeit zwischen verschiedenen Lösungsansätzen sichergestellt, da diese alle die gleichen Funktionen umsetzen müssen. Da bei der Bewertung der Interoperabilität typischerweise mehrere Funktionen betroffen sind, ist eine Bewertung auf Ebene einzelner Funktionen nicht sinnvoll. Daher werden hier jeweils einzelne Schnittstellen und in Summe letztlich die Gesamtarchitektur bewertet.

Die in der ISO/IEC 25010 vorgesehenen weiteren Dimensionen Leistungseffizienz, Portabilität, Sicherheit, Zuverlässigkeit und Wartbarkeit sind primär technische Aspekte, die nur indirekten Einfluss auf die interdisziplinäre Problemstellung der Gestaltung einer Engineering-IT-Architektur ausüben. Sie können durch IT-Spezialisten, ggf. unter Berücksichtigung einer Infrastrukturarchitektur, bewertet werden. Sie fließen daher im Rahmen dieser Arbeit nicht in das Vorgehensmodell ein, können aber bei Bedarf im Lösungsmodell als eigene Bewertungsdimensionen mit aufgenommen werden.

Strategische Bewertungskriterien: Ergänzend zu den drei o. g. Kriterien bieten die strategischen Bewertungskriterien die Möglichkeit, wichtige strategische Faktoren zu berücksichtigen. Beispiele können bereits bestehende IT-Lösungen oder Partnerschaften mit bestehenden Anbietern darstellen, die bei der Gesamtbewertung berücksichtigt werden sollen. Diese strategischen Anforderungen sind zu identifizieren und jeweils zu gewichten und für die verschiedenen Lösungsvarianten zu bewerten.

Abschließend wird ein Review der Komponentenarchitekturen jeder Lösungsvariante hinsichtlich Vollständigkeit und Korrektheit durchgeführt. Dies umfasst auch, dass alle Schnittstellen bewertet wurden. Darüber hinaus wird gepüft, ob alle Anforderungen und Anwendungsfälle bewertet wurden. Schließlich muss ebenfalls geprüft werden, dass alle variantenspezifischen Informationen der Lösungsvariante korrekt zugeordnet sind. Als **Resultat** der Phase sind alle Lösungsvarianten hinsichtlich funktionaler Eignung, Gebrauchstauglichkeit und Interoperabilität bewertet. Zudem ist für jede Lösungsvariante eine Komponentenarchitektur mit den zu nutzenden Schnittstellen definiert.

Als finaler Schritt erfolgt die **Lösungsvariantenauswahl**. Die abschließende Bewertung erfolgt im Rahmen einer Nutzwertanalyse, welche die funktionale Eignung, Gebrauchstauglichkeit, Interoperabilität und die Erfüllung strategischer Kriterien mit einbezieht (vgl. Kap. 5.3.2.7). Die Gewichtung der vier Dimensionen sollte ausgeglichen sein, kann aber situationsabhängig angepasst werden. Die beste Variante wird ausgewählt.

5.2.5 Iterativer Durchlauf des Vorgehensmodells

Die Phasen des Vorgehensmodells bauen aufeinander auf, weshalb diese sinnvollerweise in der dargelegten Reihenfolge zu erarbeiten sind. Es ist jedoch zu berücksichtigen, dass eine Vielzahl von Abhängigkeiten zwischen den Modellen existiert und Phasen bzw. Schritte daher nicht immer streng in der vorgebenen Reihenfolge durchlaufen werden. Vor- und Rücksprünge können je nach Informationslage daher durchaus sinnvoll sein und sind zulässig.

Ebenso muss berücksichtigt werden, dass ein streng lineares Vorgehen mit einmaligen Durchlaufen des Vorgehensmodells nicht zu einem vollständigen Modell führen wird. Das Vorgehen ist in Bild 5-2 zwar stringent dargestellt, die verschiedenen Aktivitäten werden in der Praxis jedoch iterativ durchgeführt. Wie in Bild 5-4 beispielhaft dargestellt, gibt es sowohl in der Analyse (Phase 1 und 2) als auch der Synthese (Phase 3 und 4) verschiedene allgemeine Faktoren, die eine Iteration auslösen können. Neben Änderungen am Zielbild können dies im Rahmen der Lösungsfindung z. B. die Aufteilung in zusätzliche Funktionen oder Änderungen an Software-Komponenten durch dessen Hersteller sein.

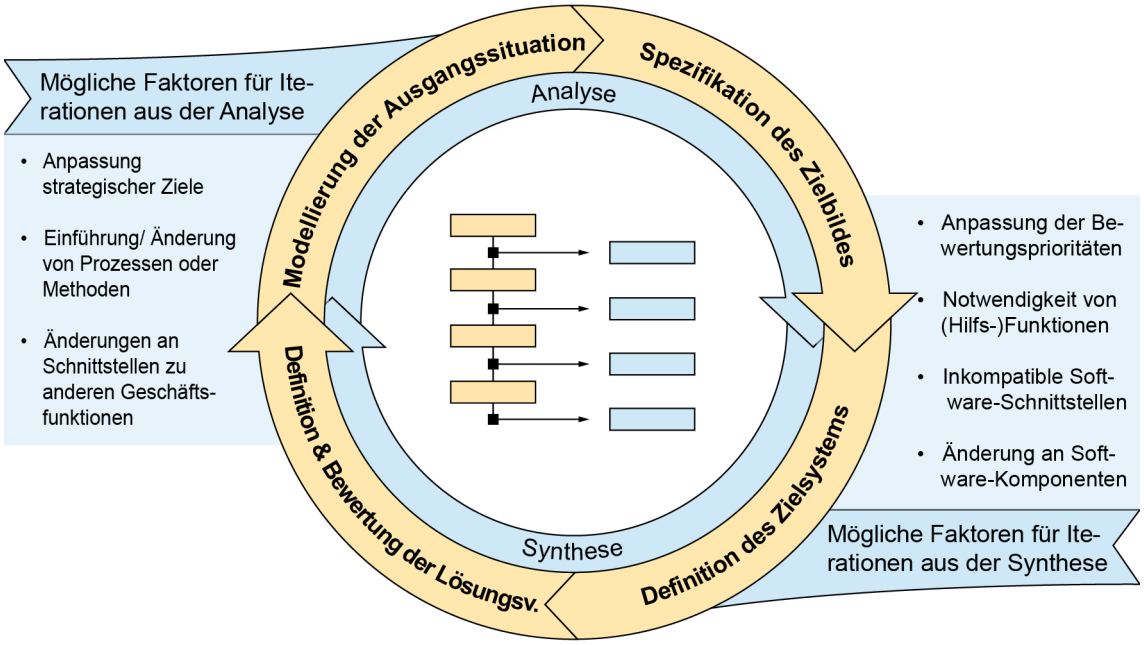


Bild 5-4: Betrachtung des Vorgehensmodells als iterativen Zyklus

5.3 Sprachkonzept

Zum Aufbau und zur Nutzung eines Modells stellt diese Arbeit ein Sprachkonzept mit einer Modellierungssprache zur Verfügung. Dieses besteht zum einen aus grundlegenden Konzepten zur Anwendung und Definition der Modellierungssprache (Kap. 5.3.1). Zum anderen umfassst dies die Erläuterung der Sprachkonstrukte und deren Semantik sowie Hinweise zur Anwendung (Kap. 5.3.2).

Die Modellierungssprache hat den Anspruch, eine ganzheitliche und domänenübergreifende Beschreibung des Engineerings zu ermöglichen. *Ganzheitlich* bedeutet in diesem Kontext, dass alle relevanten Informationen zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen enthalten sind, um ein umfassendes Verständnis hinsichtlich der Prozesse, Methoden, Artefakte und IT-Systeme zu erlangen. Eine *domänenübergreifende* Beschreibung setzt voraus, dass die Dokumentation die gemeinsamen Informationsbedarfe verschiedener Stakeholder wie Anwender, Prozess-, Methoden- oder IT-Spezialisten bedient. Sie muss daher unabhängig von den unterschiedlichen Fachgebieten der beteiligten Akteure verständlich sein.

5.3.1 Modellierungskonzept

Zunächst werden die Sprachkonstrukte in Kap. 5.3.1.1 durch ein Metamodell definiert. Kapitel 5.3.1.2 beschreibt das grundlegende Modellierungskonzept, auf dem die Sprache aufbaut, und elementare Begriffe. In Kap. 5.3.1.3 werden Sichten zur Unterstützung und Implementierung des Modellierungskonzepts vorgestellt, welche einen strukturierten Rahmen zur Erstellung und Nutzung von Modellinhalten darstellen. Das in Kap. 5.3.1.4 vorgestellte Konzept zur Rückverfolgbarkeit soll die Rückverfolgbarkeit innerhalb der Engineering-IT-Architektur sicherstellen.

5.3.1.1 Metamodell

Die abstrakte Syntax und Teile der statischen Semantik einer Modellierungssprache können, wie in Kap. 3.3.4 dargelegt, mit Hilfe eines Metamodells beschrieben werden [SVE+07, S. 28f]. Als Grundlage dient dabei das Vorgehensmodell (vgl. Kap. 5.2). Das Metamodell gibt einen Überblick über die wesentlichen Modellelemente und ihre Zusammenhänge. Die OMG sieht als Sprache zur Metamodellierung die Meta Object Facility (MOF) in der Version 2 vor. MOF selbst definiert jedoch keine eigene Notation, stattdessen wird hierzu die Notation der UML Klassendiagramme herangezogen [OMG16a-ol, S. 1f, 9]. *Assoziationen* stellen dabei eine Relation zwischen zwei Metamodellelementen dar. *Kompositionen* beschreiben eine Beziehung zwischen einem Ganzen und seinen Teilen („ist-Teil-von“). *Generalisierungen* beschreiben die Vererbung von Attributen und Beziehungen eines generellen zu einem speziellen Metamodellelement.

Das Metamodell für die Modellierungssprache der vorliegenden Spezifikationstechnik ist in Bild 5-5 dargestellt. Basierend auf dem Vorgehensmodell und der zugrundeliegenden Problemstellung wurden die entsprechenden Metamodellelemente definiert. Sie gliedern sich in drei Teilbereiche. Die Metamodellelemente der Analyse-Domäne unterstützen die Analyse der Prozesse, Methoden, Artefakte, Anwendungsfälle und weiterer relevanter Objekte. Die Elemente der Lösungs-Domäne erlauben die Beschreibung von funktionalen oder Softwarearchitekturen und den dazu notwendigen Elementen und Verbindungen. Im Bereich der Anforderungen & Rückverfolgbarkeit sind Anforderungen und Beziehungen zur Sicherstellung der Rückverfolgbarkeit definiert.

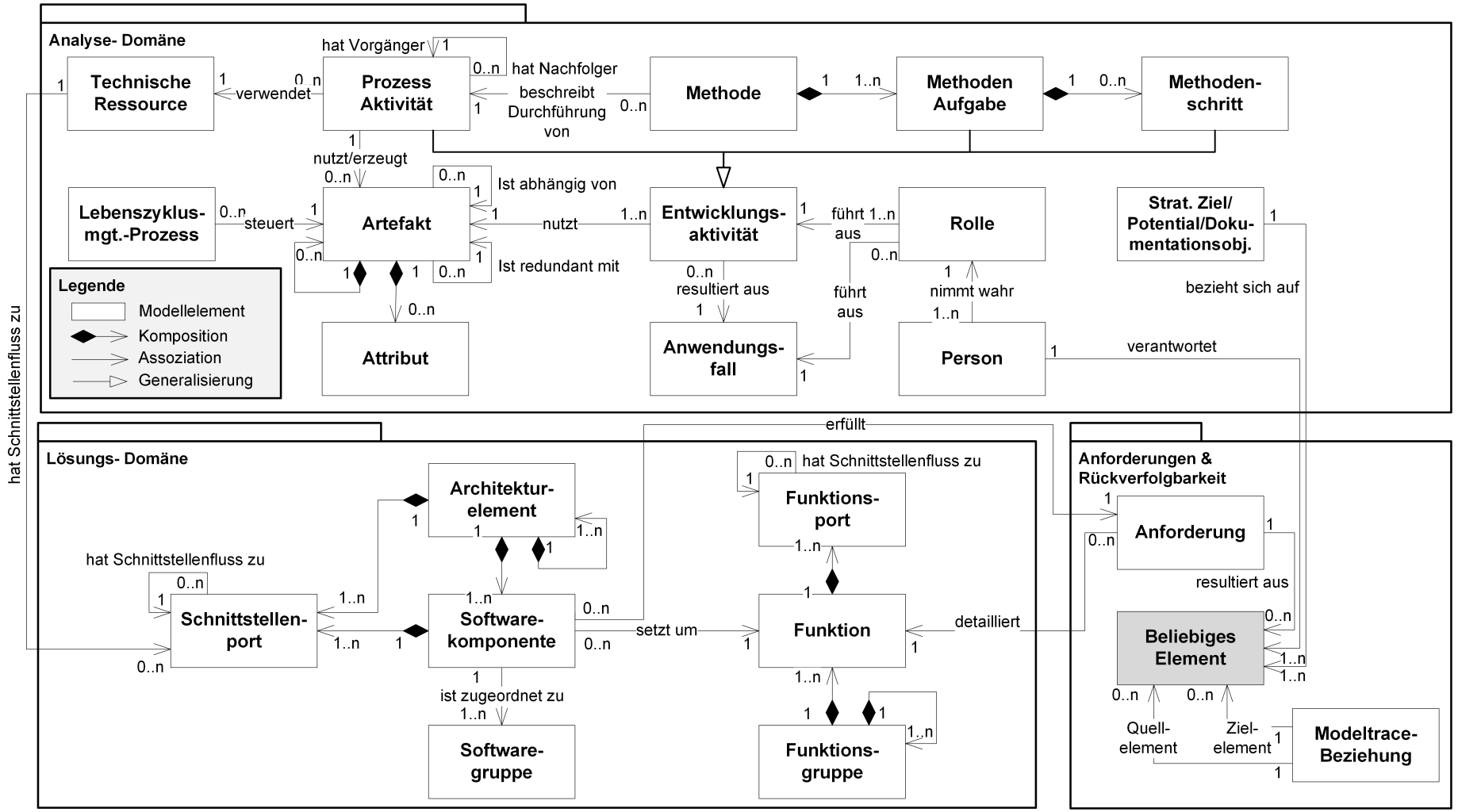


Bild 5-5: Metamodell der Modellierungssprache

So kann bspw. dem Metamodell entnommen werden, dass ein Anwendungsfall eine „resultiert aus“-Beziehung zu allen drei Arten von Entwicklungsaktivitäten aufweisen kann und durch eine „führt aus“-Beziehung mit einer Rolle verbunden sein kann. Die Semantik der jeweiligen Metamodellelemente und ihre Verwendung im Rahmen der Modellierung ist im Rahmen der Partialmodelle in Kap. 5.3.2 detailliert textuell beschrieben.

5.3.1.2 Grundlagen des Modellierungskonzepts

Die Modellierung mit der vorgestellten Modellierungssprache basiert auf dem Prinzip der **Trennung zwischen Modell und Sicht**. Es ist in Bild 5-6 dargestellt, basiert auf der ISO/IEC/IEEE 42010 und ist insbesondere bei der rechnergestützten Modellierung verbreitet. Das **Modell** stellt die Summe aller Modellelemente dar, welche die Engineering-IT-Architektur beschreiben – im Kern sind dies Architekturelemente, ihre Attribute und Beziehungen. Um das Modell sowohl als Mensch, als auch rechnergestützt nutzen zu können, ist es notwendig, dass die abstrakte Syntax implementiert und den Modellelementen zugeordnet ist. Bereits der Umfang eines Modells von moderat-komplexen Sachverhalten kann nicht vollständig durch einen Menschen überblickt und interpretiert werden. Bei der rechnergestützten Modellierung empfiehlt es sich daher, sämtliche Modellelemente maschinenverarbeitbar abzulegen, wozu man sich eines sog. **Modellrepositorys** bedient. Dieses erfüllt somit eine Speicherfunktion, typischerweise in Form einer Datenbank. Die Modellelemente sind dort als Datenobjekt abgelegt.

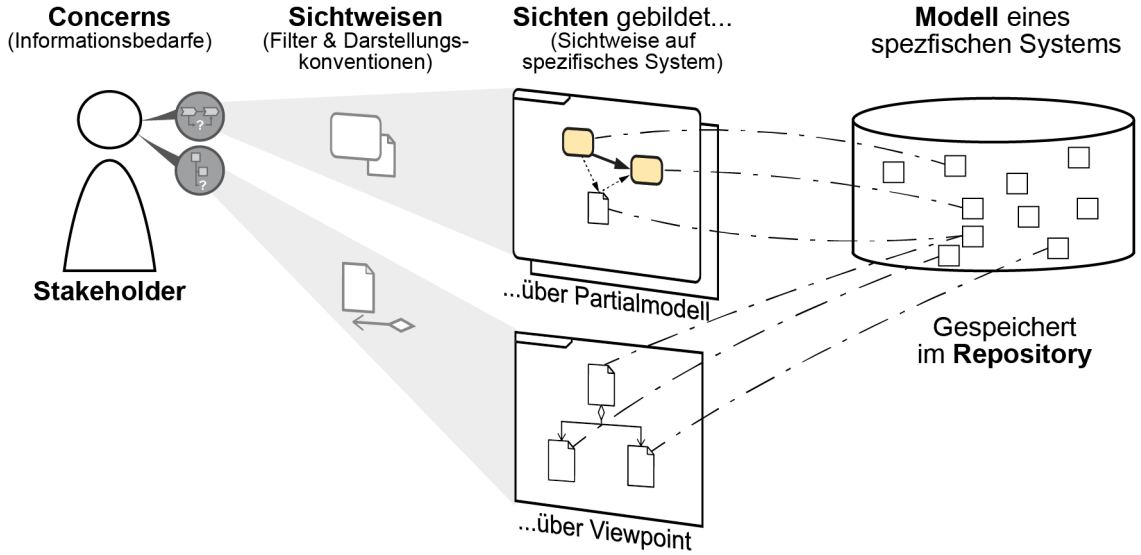


Bild 5-6: Die Trennung zwischen Modell und Sicht als wesentliche Grundlage der Spezifikationstechnik in Anlehnung an [Alt12, S. 39]

Um als Mensch die Informationen bedarfsgerecht verarbeiten zu können, werden *Sichten* auf das Modell genutzt. Eine Sicht stellt einen spezifischen Ausschnitt des Modells dar, um einen Informationsbedarf (Concern, vgl. Kap. 4.1.1) zu befriedigen. In der Praxis werden Sichten typischerweise über Diagramme dargestellt. In diesen Diagrammen wird oftmals auch erst die konkrete Syntax sichtbar. Durch die Trennung von Modellelement und

dessen Darstellung ist es auch möglich, das gleiche Modellelement in mehreren Sichten darzustellen. Es ist jedoch auch denkbar, dass ein Modellelement in keiner Sicht auftaucht. Das systematische Arbeiten mit Sichten setzt Sichtweisen voraus. Eine **Sichtweise** ist eine Art, Systeme zu betrachten. Sie definiert die notwendigen Konventionen, um eine Architektur im Sinne der Sichtweise darzustellen [ISO42010, S. 20]. Eine Sicht ist somit die Anwendung einer Sichtweise auf ein spezifisches System.

Beispielsweise können in einem Modell alle Prozesse, Methoden, Anwendungsfälle und Anforderungen enthalten sein, die für die Umsetzung von Systems Engineering in einem spezifischen Unternehmen notwendig sind. Da dies jedoch zu tausenden Modellelementen führen kann, wäre die Darstellung aller Elemente in einer Sicht nicht sinnvoll. Um sinnvolle Sichten zu bilden, werden nun Sichtweisen herangezogen. Eine mögliche Sichtweise könnte sein, nur alle Anforderungen darzustellen, die einer bestimmten Funktion zugeordnet sind. Wendet man diese Sichtweise auf ein konkret vorliegendes Modell an, entsteht die Sicht. Sie enthält bspw. alle Anforderungen für die Funktion „Anforderungen handhaben “.

5.3.1.3 Sichtenkonzept

Um die Arbeit mit dem Modell zu erleichtern und das Prinzip der Trennung zwischen Modell und Sicht systematisch zu unterstützen, wurden im Rahmen der Arbeit zwei Arten von Sichtweisen definiert, welche in Bild 5-7 dargestellt sind. Die sieben **Partialmodelle** dienen primär dem Zweck, Informationen im Modell zusammenhängend zu erfassen. Nach dieser Betrachtung besteht die Engineering-IT-Architektur im Kern aus einem System kohärenter Partialmodelle. Neben den Partialmodellen existieren die Viewpoints. Die 34 **Viewpoints** unterstützen primär die Analyse und Detaillierung des Modells. Dazu sind sie deutlich fokussierter als die Partialmodelle geschnitten. Viewpoints weisen Schnittmengen mit den Partialmodellen auf, sind jedoch nicht 1:1 zuordbar (s. Anhang A1.2). Partialmodelle und Viewpoints sind demnach zwei koexistente Arten von Sichtweisen, die je nach Bedarf und Problemstellung parallel zueinander genutzt werden können.

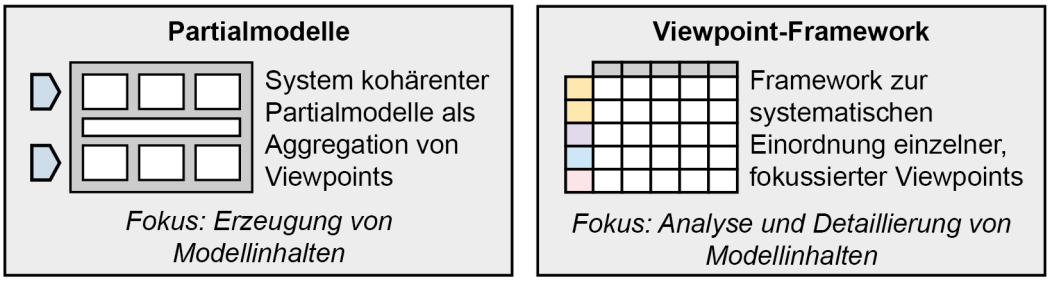


Bild 5-7: Überblick über die Arten von Sichtweisen auf das Modell

Es existieren sieben **Partialmodelle**, um den Kern der Engineering-IT-Architektur zu beschreiben. Die Partialmodelle orientieren sich hinsichtlich ihres Umfangs am methodischen Vorgehen. Sie unterstützen insbesondere die Erstellung des grundlegenden Modells im Durchlauf des Vorgehensmodells. Die Partialmodelle betrachten nur ausgewählte Arten von Metamodellelementen und bieten somit eine domänen- bzw. problemorientierte Sicht

auf das Modell. Bild 5-8 stellt die Partialmodelle in der Übersicht dar, welche nachfolgend kurz beschrieben werden.

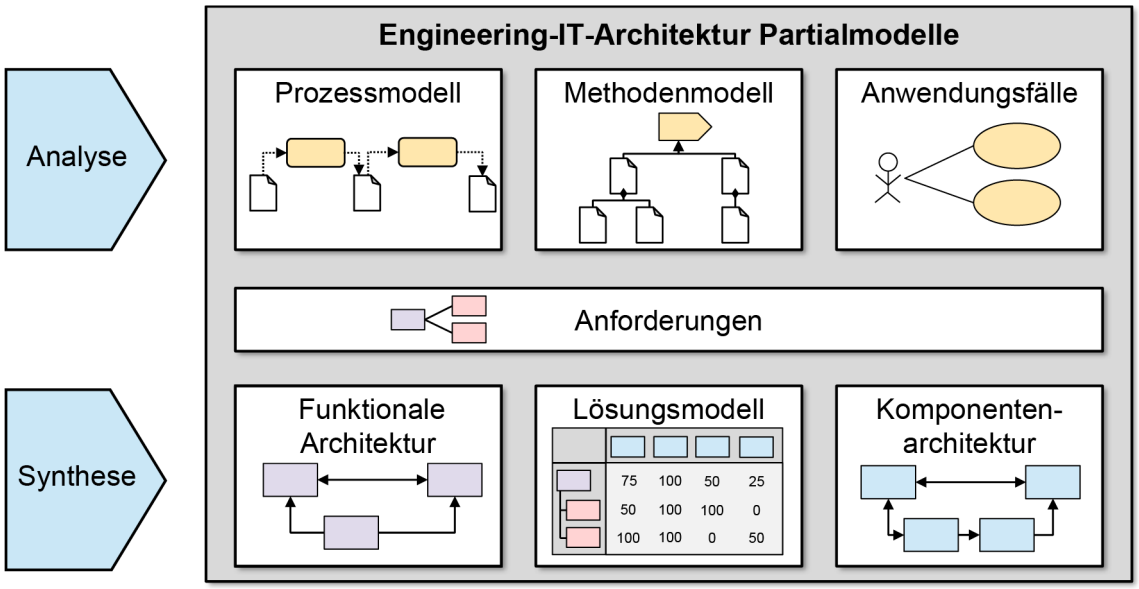


Bild 5-8: Überblick über die Partialmodelle der Spezifikationstechnik

- **Prozessmodell:** Das Prozessmodell beschreibt die Geschäftsprozesse und die Prozessaktivitäten sowie als In- oder Output genutzte Artefakte. Dabei steht die sachlogische Abfolge im Vordergrund. Darüber hinaus werden den Prozessaktivitäten genutzte Methoden oder technische Ressourcen zugeordnet.
- **Methodenmodell:** Die identifizierten Methoden werden jeweils durch relevante Tätigkeiten beschrieben, die Abfolge ist dabei sekundär. Das Methodenmodell greift die Artefakte des Prozessmodells auf und kann diese hierarchisch detaillieren, bei Bedarf bis auf Attributebene. Zudem können Beziehungen zwischen Artefakten dargestellt und redundante Artefakte zwischen Methoden markiert werden. Diese Erkenntnisse können so systematisch aus den Methoden abgeleitet werden.
- **Anwendungsfälle:** Die Anwendungsfälle beschreiben die erwartete Werkzeugunterstützung aus Anwendersicht zur Durchführung der Tätigkeiten aus dem Prozess- und Methodenmodell. Sie beschreiben reproduzierbar typische und oft wiederkehrende Situationen im Werkzeug und dienen daher auch zum Test der Gebrauchstauglichkeit.
- **Anforderungen:** Die im Rahmen aller Partialmodelle abgeleiteten Anforderungen werden im Anforderungsmodell gesammelt. Als übergeordnete Struktur dienen die Funktionen.
- **Funktionale Architektur:** Durch die IT-Infrastruktur bereitzustellende Funktionen werden als hierarchische Struktur beschrieben. In diese werden zur Erstellung des Zielsystems Anforderungen und Anwendungsfälle eingeordnet. Im Rahmen der Architektur werden sie mit ihren Beziehungen zum Informationsaustausch dargestellt.

- **Lösungsmodell:** Das Lösungsmodell umfasst geeignete Lösungsvarianten, welche auf verschiedenen, konsistenten Kombinationen von Softwarekomponenten basieren. Für jede Lösungsvariante werden differenziert die Bewertungsergebnisse der ganzheitlichen Lösung modelliert.
- **Komponentenarchitektur:** Die Softwarekomponenten einer spezifischen Lösungsvariante sind miteinander in Beziehung gesetzt. Der Fokus des Modells liegt auf den Schnittstellen zwischen den Softwarekomponenten.

Die Partialmodelle eignen sich insbesondere, um entlang des Vorgehensmodells die Modellinhalte zu erzeugen. Die detaillierte Modellierung mit Hilfe der Partialmodelle ist in Kap. 5.3.2 beschrieben. Sie weisen dazu einen Kompromiss zwischen Fokussierung und Breite der Sichtweise auf. Um jedoch ein bestehendes Modell differenzierter zu analysieren oder zu detaillieren, bieten sich die **Viewpoints** an. Viewpoints beschreiben ebenso wie die Partialmodelle eine Sichtweise, sind dabei jedoch deutlich differenzierter. Sie eignen sich daher bspw. besonders, um fehlende Informationen im Modell zu identifizieren, Beziehungen zwischen den Partialmodellen zu ergänzen oder partialmodellübergreifende Aspekte zu modellieren oder analysieren.

Die Viewpoints sind in im sogenannten **Viewpoint-Framework** definiert. Dieses ist in Tabelle 5-1 dargestellt und durch eine zweidimensionale Matrix strukturiert. Die Spalten beschreiben einzelne Aspekte des betrachteten Systems: Den Kontext, das Verhalten & durchzuführende Tätigkeiten, die Struktur & Objekte, Schnittstellen & Interaktionen sowie die Traceability, Mappings und das Management zwischen Modellelementen. Die Zeilen fokussieren die verschiedenen Domänen und Problemstellungen bei der Entwicklung einer Engineering-IT-Architektur. Dazu gehören Engineering Prozesse, Methoden, notwendige Funktionen, IT-Applikationen und resultierende Anforderungen. Innerhalb der Zellen sind die einzelnen Viewpoints aufgeführt. Jeder Viewpoint besitzt zur eindeutigen Identifizierung einen alphanumerischen Bezeichner, z. B. PFV04a. Der zweite Buchstabe markiert dabei die Zeile, die Zahl die Spalte. Existieren mehr als zwei Viewpoints in einer Zelle, werden diese durch den angehängten Kleinbuchstaben differenziert. Die jeweiligen Viewpoints sind in Anhang A2 durch eine Kurzbeschreibung, die dargestellten Modellelemente und typische Darstellungsformen detailliert beschrieben.

Das Viewpoint-Framework erlaubt es somit, auf systematisch definierte und strukturierte Sichten zuzugreifen. Es unterstützt somit die Standardisierung der Sichten im Modell. Zudem trägt die Ontologie des Frameworks dazu bei, keine relevanten Modellaspekte zu übersehen und die Vollständigkeit des Modells differenzierter und aus verschiedenen Perspektiven bewerten zu können. Sollte im Rahmen des Vorgehens Bedarf bestehen, einzelne Aspekte fokussiert zu betrachten, kann mittels des Frameworks ein geeigneter Viewpoint ausgewählt werden. Beispielsweise erlauben einzelne Viewpoints die Betrachtung aller Tätigkeiten einer Methode oder alternativ aller Artefakte und deren Struktur. Sie erlauben somit eine feinere Untergliederung der Partialmodelle. Es können jedoch auch partialmodellübergreifende Viewpoints genutzt werden, z. B. die Rollen aller Prozess-, Methoden-

Tabelle 5-1: Sichtenkonzept mit den im Grid eingeordneten Sichtweisen (Viewpoints)

	Kontext	Verhalten & Tätigkeiten	Struktur & Objekte	Schnittstellen & Interaktionen	Traceability, Mappings & Management
Prozess Domäne	PPV 01 Organisations Umfeld	PPV 02 Prozessablauf	PPV 03a Artefaktliste PPV 03b Ressourcenliste	PPV 04a Prozess-schnittstellen PPV 04b Prozess-potentiale	PPV 05 Lebenszy-klusmgt.-Mpg
Methoden Domäne		PMV 02 Methoden-diagramm	PMV 03 Artefakt-dekomposition	PMV 04a Aktivitäts-Methoden-Mpg. PMV 04b Artefakt-beziehungen PPV 04c Methoden-potentiale	
Funktions Domäne	PFV 01 Funktions Umfeld		PFV 03 Funktionale Dekomposition	PFV 04a Funktionale Architektur PFV 04b Funktionale Schnittstellen	PFV 05 Funktions-Tra-ceability-Mpg.
IT-Applikations Domäne	PAV 01 Applikations Umfeld		PAV 03a Architektur Dekomposition PAV 03b SW-Komponen-ten-Zuordnung	PAV 04a Applikations-architektur PAV 04b Software Schnittstellen	PAV 05a SW-Funkti-ons-Mpg. PAV 05b Variantendefini-tion & -mgt.
Anforderungs Domäne	PRV 01a Rollen-identifikation PRV 01b Personen-identifikation PRV 01c Strategische Ziele	PRV 02 Anwendungs-fälle	PRV 03a Anf.-Funktions-zuordnung PRV 03b AnwFall-Funk-tionszuordnung PRV 03c Potentialliste		PRV 05a Anf.-Traceabili-ty-Mpg. PRV 05b Lösungs-bewertung PRV 05c Rollen-Perso-nen-Mpg.

Legende Sichtweise (Viewpoint) Mpg.: Mapping Mgt.: Management

und Anwendungsfallmodelle. Die detaillierten Schnittmengen zwischen Partialmodellen und Viewpoints sind in Anhang A1.2 dargestellt.

5.3.1.4 Vernetzung der Partialmodelle

Um das Vorgehensmodell und das modellbasierte Arbeiten zu unterstützen, sieht die Spezifikationstechnik vor, Modellelemente miteinander zu verknüpfen. Erstens kann dies innerhalb einer Sichtweise stattfinden, z. B. innerhalb einer Hierarchie homogener Modellelemente. Zweitens können Modellelemente durch die Trennung zwischen Modell und Sicht zwischen den Sichtweisen miteinander verknüpft werden. Tabelle 5-2 gibt dazu einen Überblick über die wesentlichen Zusammenhänge zwischen den Modellelementen. Dabei unterscheidet die Matrix zwischen Kompositionsbeziehungen (Hierachische Struktur zwischen zwei Modellelementen), Fluss-Beziehungen (Austausch von Daten zwischen zwei Modellelementen), Evaluierungsbeziehungen (Allokation von Lösungselemten auf Funktionen), Containment-Beziehungen (Hierarchie in der Paketstruktur des Modells), Modell-Trace-Beziehungen (Rückverfolgbarkeit zwischen Modellelementen) sowie Assoziations-Beziehungen (alle anderen Beziehungen; Abhängigkeit zwischen zwei Model-

lementen)[4]. Die Matrix gibt von einer Zeile zur Spalte gelesen die Richtung der Beziehung wieder. So kann der Matrix z. B. entnommen werden, dass eine Assoziations-Beziehung von der Methode zu einer Aktivität eines Prozesses erstellt wird (Zeile 3/Spalte 2).

Tabelle 5-2: Relevante Beziehungen zur sichtenübergreifenden Vernetzung von Modellelementen

Beziehungsmatrix **Fragestellung:** „Welche Beziehung kann zwischen Elementtyp i (Zeile) und Elementtyp j (Spalte) sinnvollerweise existieren?"		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
		Swimlane	**Aktivität**	**Methode**	**Aufgabe**	**Methoden-Schritt**	**Artefakt**	**Attribut**	**Rolle**	**Person**	**Potential/Strat. Ziel/Doku.**	**Lebenszyklusmgt-Prozess**	**Technische Ressource**	**Umfeldelement**	**Austauschartefakt**	**Anwendungsfall**	**Funktionsgruppe**	**Funktion***	**(Ergänz.) Anforderung**	**Softwaregruppe**	**Softwarekomponente***	**Teilarchitektur***
1															(R)							
2			R				R								R							
3			R		P	P	P	P			P											
4						A																
5																						
6			R		R	R	A/R	A														
7			R		R	R	R	R														
8			R		R	R	R	R	A		R	R				R						
9									R													
10			R	R	R	R	R	R	R	R	A	R	R	R	R	R	R	R	R	R	R	R
11			A				R	R														
12			R		(R)	(R)											C	C			C	C
13															R							
14		(R)	R											R								
15			X		X	X	X	X	R/X	X	X	X	X	X	X								
16			X		X	(X)	X	(X)	X	X	X	X	C/X	X	X	X	A/C	A/C					
17			X		X	X	X	X	X	X	X	X	C/X	X	X	X	C	C	P			
18			X		X	X	X	X	X	X	X	X	X	X	X	X					X	
19																	X	X	X		A	
20													C						E		C	C
21													C								A/C	C

Legende A: Kompositions-Bez. C: Fluss-Bez. E: Evaluierungsbez. P: Containment -Bez.
R: Assoziations-Bez. X: Model-Trace-Bez. (): Beziehung kontextabhängig möglich *: Bei Instanzen nur „C"

[4] Die verschiedenen Arten von Assoziationsbeziehungen sind zur Wahrung der Übersichtlichkeit nicht in Tabelle 5-2 beschrieben, jedoch in Kap. 5.3.2.

Bedingt durch das Vorgehensmodell bauen die Modelle aufeinander auf. Dies betrifft auch die Beziehungen der Modellelemente, welche im Folgenden basierend auf den Partialmodellen skizziert werden sollen. Das **Prozessmodell** stellt hinsichtlich des Detailgrades das gröbste Modell dar und dient damit als Referenz. Im Prozessmodell identifizierte Methoden werden im Rahmen des **Methodenmodells** detailliert. Aus dem Prozess- und Methodenmodell werden durch ein IT-Werkzeug zu unterstützende, wiederkehrende Tätigkeiten abgeleitet, die durch die **Anwendungsfälle** beschrieben werden. Da insbesondere die **Anforderungen** kontinuierlich über das gesamte Vorgehensmodell hinweg erhoben werden, können sie prinzipiell aus allen anderen Partialmodellen abgeleitet werden, obgleich der Schwerpunkt auf den lösungsneutralen Partialmodellen liegt. Die Funktionen der **funktionalen Architektur** können aus den Anforderungen abgeleitet werden. Zudem werden die Anforderungen und die Anwendungsfälle den Funktionen zugeordnet, um ein hierarchisches Zielsystem aufzubauen. Dieses wird durch das **Lösungsmodell** repräsentiert, indem es Lösungsvarianten definiert und für diese Anforderungen und Anwendungsfälle auf Softwarekomponenten allokiert und bewertet. Die Softwarekomponenten werden im Rahmen der **Komponentenarchitektur** miteinander in Beziehung gesetzt.

Die Nutzung der Beziehungen innerhalb und zwischen den Sichtweisen ist in Kap. 5.3.2 auf Basis der Partialmodelle detailliert beschrieben. Sie wird durch spezifische Viewpoints ergänzt, insbesondere hinsichtlich Traceability, Mappings & Management.

5.3.2 Sprachkonstrukte und ihre Anwendung

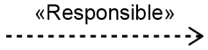
Dieses Kapitel stellt die Partialmodelle und Sprachkonstrukte detailliert vor. Zunächst werden in Kap. 5.3.2.1 übergreifende Sprachkonstrukte eingeführt, welche in verschiedenen Partialmodellen nutzbar sind. In Kap. 5.3.2.2 werden als Grundlage der *Analysephase* zunächst die Prozesse zur Orientierung abgebildet und analysiert. Darauf aufbauend werden die Methoden in Kap. 5.3.2.3 modelliert und Artefakte sowie ihre Beziehungen detailliert. Die in Kap. 5.3.2.4 beschriebenen Anwendungsfälle ergänzen typische Interaktionen mit den Werkzeugen aus Anwendersicht. Die Entwicklung einer für die *Synthesephase* notwendigen funktionalen Architektur wird in Kap. 5.3.2.5 vorgestellt. Sie beschreibt lösungsneutrale Funktionen und ihre Beziehungen auf Grundlage der Anforderungen. In Kap. 5.3.2.6 werden die Anforderungen erläutert, die mit allen weiteren Partialmodellen eng verknüpft sind. Kap. 5.3.2.7 beschreibt das Lösungsmodell, welches den Funktionen systematisch geeignete Softwarekomponenten zuordnet sowie hinsichtlich der Anforderungserfüllung bewertet und dadurch konsistente Lösungsvarianten identifiziert. Die Lösungsvarianten werden parallel dazu wie in Kap. 5.3.2.8 als Komponentenarchitektur modelliert, welche die Softwarekomponenten und ihre Beziehungen beschreiben.

5.3.2.1 Übergreifende Sprachkonstrukte

Einige Sprachkonstrukte lassen sich in verschiedenen Partialmodellen nutzen und werden daher als übergreifende Sprachkonstrukte bezeichnet. Diese werden im Folgenden einge-

führt und können im Rahmen der Partialmodelle dann nach Bedarf genutzt werden. Sie sind in Tabelle 5-3 dargestellt[5].

Tabelle 5-3: Übersicht über die übergreifenden Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Rolle «Role»	Name	Beschreibt eine temporäre, durch Aufgaben, Befugnisse und Verantwortungen abgrenzbare Funktion, welche durch eine Person ausgeübt werden kann.
Person «Person»	Name	Beschreibt eine spezifische und konkret benannte Person, welche eine oder mehrere Rollen ausüben kann.
Potential «Potential»	Name	Beschreibt eine Schwachstelle bzw. ein Verbesserungspotential hinsichtlich der Engineering-IT-Architektur.
Strategisches Ziel «StrategicGoal»	Name	Beschreibt durch eine übergeordnete Strategie vorgegebene Zielsetzung, die durch die Engineering-IT-Architektur umgesetzt werden soll.
Dokumentationsobjekt «Documentation»	Name	Identifiziert ein Dokument, welches Prozesse, Methoden oder andere relevante Modellelemente beschreibt (z.B. ein Methodenleitfaden).
Assoziation *Association (UML)*		Verknüpft zwei Modellelemente durch eine gerichtete Beziehung.
Verantwortlichkeit «Responsible»	«Responsible» 	Verknüpft eine Person mit einem Modellelement, um zu dokumentieren, dass diese für die fachliche Definition und Ausgestaltung des Elementes verantwortlich ist.
Modell-Trace «ModelTrace»	«ModelTrace»	Stellt die Rückverfolgbarkeit zwischen zwei Elementen des Modells sicher und wird vom Ziel (z.B. Anwendungsfall) zur Quelle (z.B. Methodenschritt) erstellt.

Zu den übergreifenden Sprachkonstrukten gehören Rollen und Personen. Eine **Rolle** ist eine temporäre, durch Aufgaben, Befugnisse und Verantwortung abgrenzbare Funktion. Zwei Beispiele stellen die Rollen des Produktmanagers oder Systemingenieurs dar. Eine Rolle wird durch einen Titel und eine textuelle Kurzbeschreibung als Attribut definiert. Sie kommt in Prozess- und Methodenmodellen sowie Anwendungsfällen zum Einsatz und wird dort per Assoziation mit den von ihr durchgeführten oder verantworteten Tätigkeiten verknüpft. Eine Rolle kann durch eine konkrete **Person** ausgeübt werden, z. B. Max Mustermann. Sie wird per Assoziation mit der Rolle verknüpft. Dies erlaubt z. B., konkrete Ansprechpartner zur Informationsbeschaffung ausfindig zu machen.

Potentiale sind Schwachstellen bzw. Verbesserungspotentiale hinsichtlich der Engineering-IT-Architektur. Sie können in allen Partialmodellen erstellt werden und einem beliebigen Modellelement per Assoziation zugeordnet werden. Die Beschreibung erfolgt über einen

[5] In den Tabellen zur Erläuterung der Sprachkonstrukte sind in der ersten Spalte jeweils in mager gesetzter Schrift die Stereotypen des STEITA-Profils in «Guillemets» eingefasst oder in kursiver Schrift die Metaklassen bzw. Stereotypen der UML, SysML, SPEM oder BPMN angegeben.

Titel sowie per textueller Kurzbeschreibung und Prioritätskennzahl als Attribut. Ein Beispiel stellt das Potential „Fehlende Rückverfolgbarkeit zwischen Anforderungen und Architektur“ dar.

Strategische Initiativen sind oft der Auslöser zur Umgestaltung einer Engineering-IT-Architektur. Ein **strategisches Ziel** beschreibt einen übergeordneten und durch die Prozess- oder Systemebene des 4-Ebenen-Modells umzusetzenden SOLL-Zustand, der vom aktuellen IST-Zustand abweicht. Es wird durch einen Titel und eine textuelle Beschreibung spezifiziert und kann in allen Partialmodellen genutzt werden. Um Modellelemte zur Rückverfolgbarkeit miteinander zu verknüpfen, wird eine **Modell-Trace-Beziehung** genutzt.

Zur Erstellung der Engineering-IT-Architektur werden häufig bestehende Dokumente wie Prozess- oder Methodenleitfäden herangezogen, aus denen die Partialmodelle abgeleitet werden. Damit die Modellelemente auch zu diesen Quellen rückverfolgbar sein können, gibt es sogenannte Dokumentationsobjekte. Ein **Dokumentationsobjekt** identifiziert ein oder mehrere Dokumente. Beispiele sind unter anderem Leitfäden, Prozess- und Methodenbeschreibungen, Checklisten oder Vorlagen. Die Rückverfolgbarkeit auf ein Dokumentationsobjekt wird über eine Modell-Trace-Beziehung sichergestellt.

Da sich die Engineering-IT-Architektur später aus einer Vielzahl von Architekturelementen zusammensetzt, müssen Verantwortlichkeiten für die Bestandteile der Architektur klar sein. Die **Verantwortlichkeits-Beziehung** erlaubt es, einem beliebigen Modellelement in jedem Partialmodell eine für dessen fachliche Definition und Ausgestaltung verantwortliche Person zuzuweisen. Dies kann bspw. ein Domäneverantwortlicher sein, der für eine Methode die notwendigen Aktiväten, Artefakte und deren Beschaffenheit definiert. Für alle Modellelemente muss dabei die Verantwortlichkeit klar ersichtlich sein, durch eine direkte Verknüpfung des Modellelements oder dessen hierarchisch übergeordneten Elements zu einer Person.

5.3.2.2 Prozessmodell

Das Prozessmodell beschreibt Prozesse in Form von sachlogischen Abläufen, bestehend aus Aufgaben, ihren In- und Outputs und ggf. ergänzenden Informationen wie genutzten Methoden oder technischen Ressourcen. Dazu werden zunächst mittels des Kontextmodells in einer Black-Box Perspektive das Umfeld des zu betrachtenden Prozesses und die relevanten Schnittstellen betrachtet. Nachfolgend kann durch das interne Prozessmodell die White-Box Perspektive eingenommen und der Prozess ausdetailliert werden.

Kontextmodell

Das Kontextmodell dient der Identifizierung von Anforderungsquellen und der Definition der Systemgrenze zur Abgrenzung des Engineerings. Dabei wird das untersuchte Engineering-System als Black-Box betrachtet. Es werden alle In- und Outputs des Engineering-Systems beschrieben, die mit Umfeldelementen auf prozessualer Ebene in

Beziehung stehen. Umfeldelemente sind Stakeholder, oft in Form von Organisationseinheiten oder -funktionen, die mit dem Engineerig in Beziehung stehen. Die Beziehung wird näher spezifiziert, z. B. hinsichtlich ausgetauschter Informationen. Das Kontextmodell geht somit auf den Umstand einer stärkeren Vernetzung und Durchgängigkeit von Daten und Informationen im Rahmen der Digitalisierung ein. Tabelle 5-4 fasst die wesentlichen Konstrukte und ihre Bedeutung zusammen.

Tabelle 5-4: Übersicht über die für das Kontextmodell relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Umfeldelement *Pool (BPMN)*	Name	Repräsentiert einen Bereich einer Organisation, der Beziehungen zum Engineering aufweist, jedoch außerhalb dessen Systemgrenze liegt.
Engineering System *Pool (BPMN)*	Name	Repräsentiert das Engineering als Betrachtungsgegenstand und grenzt dieses gegenüber dem Umfeld ab.
Austauschartefakt «ExchangeArtefact»		Spezifiert ein Artefakt, welches zwischen dem Engineering und Umfeldelementen über einen Artefaktaustauschfluss ausgetauscht wird.
Artefaktaustauschfluss *MessageFlow (BPMN)*		Beschreibt Quelle und Ziel eines Austausch von Artefakten zwischen dem Engineering und einem Umfeldelement. Kann über Austauschartefakte detailliert werden.

Das Kontextmodell basiert auf dem Kollaborationsdiagramm der BPMN. Das **Engineering-System** wird durch einen Pool dargestellt, der später die Prozesse des Engineerings umfasst und dessen Rahmen die Systemgrenze darstellt. Sie grenzt das zu analysierende und gestaltende Engineering gegenüber dem Umfeld ab, auf welches typischerweise kein Einfluss genommen werden kann. Die **Umfeldelemente** werden als grau ausgefüllte Pools repräsentiert, um eine optische Abgrenzung zu unterstützen. Es werden alle Umfeldelemente modelliert, die eine Schnittstelle zum Engineering ausweisen. Diese können auch Bereiche außerhalb des Unternehmens umfassen.

Über die Schnittstellen werden Informationen zwischen dem Engineering und einem Umfeldelement über **Artefaktaustauschflüsse** ausgetauscht. Sie enden jeweils an den Grenzen der Pools. Dem Artefaktaustauschfluss wird ein **Austauschartefakt** zugewiesen, welches die ausgetauschten Informationen näher spezifiziert. Dazu kann das Austauschartefakt durch verschiedene steckbriefartige Attribute beschrieben werden. Diese erlauben bei der frühen Schnittstellenidentifikation eine semiformale Beschreibung der konstituierenden Merkmale. So kann neben einer Kurzbeschreibung der Inhalt beschrieben werden, also, was für Informationen ausgetauscht werden sollen. Darüber hinaus können Anforderungen hinsichtlich der Struktur textuell beschrieben und charakterisiert werden. Weiterhin können im Umfeldelement verarbeitbare Datenformate angegeben werden, welche später eine grobe Abschätzung hinsichtlich der IT-Schnittstellen erlauben. Zudem werden die Schnittstellen bezüglich ihrer Interoperabilität beschrieben. Der Automatisierungsgrad des

Datenaustausch wird dazu in vier Stufen in Anlehnung an das NATO NC3TA Reference Model for Interoperability definiert (vgl. [SB07, S. 4], [TOG18, S. 245f]).

- **Unstrukturierter Datenaustausch** (Stufe 1): Es findet kein automatisierter Datenaustausch statt. Ausgetauschte Daten sind unstrukturiert und durch Menschen interpretierbar, z. B. ein Fließtext, der einen Anwendungsfall beschreibt oder eine Architektur als Pixelgrafik.
- **Strukturierter Datenaustausch** (Stufe 2): Die Daten folgen einer definierten Struktur und sind durch Menschen interpretierbar. Sie können manuell und/oder automatisiert verarbeitet werden, der Verarbeitungsvorgang erfordert jedoch eine manuelle Handlung. Ein typisches Beispiel ist der datei-basierte Austausch über Im-/Exportfunktionen, z. B. der Austausch von Anforderungen über ein tabellarisch strukturiertes Dokument, bei dem jede Zeile eine Anforderung und die Spalten ihre Attribute darstellen. Der Import und das Mapping der Spalten auf die Attribute im Zielwerkzeug erfordern jedoch ein manuelles Handeln.
- **Nahtloser Austausch von Daten** (Stufe 3): Daten können automatisiert zwischen IT-Werkzeugen auf Grundlage eines gemeinsamen Austauschmodells ohne manuelle Handlung ausgetauscht werden. Die Daten müssen daher einer definierten Struktur folgen und sind rechnerverarbeitbar. Beim Datenaustausch müssen ggf. Einschränkungen in Kauf genommen werden, z. B. der teilweise Verlust von Informationen, kein Echtzeit-Datenaustausch, Einschränkung der Bidirektionalität, etc. Ein Beispiel stellt die Integration einer Architektur in visueller Notation in eine Anforderungsspezifikation als Pixelgrafik dar, welche mehrmals am Tag automatisch aktualisiert wird. Die Architektur kann jedoch aus dem RE-Werkzeug nicht bearbeitet werden, Meta-Daten der Architekturelemente sind dort ebenfalls nicht verfügbar.
- **Nahtloser Austausch von Informationen** (Stufe 4): Die höchste Stufe ermöglicht eine einheitliche Interpretation von Informationen zwischen mehreren IT-Werkzeugen durch gemeinsame Datenverarbeitung. Die Daten sind ebenso wie in Stufe 3 strukturiert und computerverarbeitbar, erweitern diese jedoch durch Minimierung der Einschränkungen beim Datenaustausch, die die erforderliche Interoperabilität nicht beeinträchtigen. Ein Beispiel stellt die Integration eines Architekturdiagramms in einer Anforderungsspezifikation dar, wobei die Meta-Daten der Architekturelemente innerhalb des RE-Werkzeugs manipuliert werden können. Die Änderungen sind daraufhin auch im Architekturmodellierungswerkzeug ersichtlich.

Für die Erstellung des Kontextmodells werden zunächst potentielle Umfeldelemente identifiziert. Dazu können bspw. Organigramme, Projektmanagementhandbücher oder Stakeholderanalysen herangezogen werden. Weiterhin können Umfeldelemente im Rahmen eines ersten Interviews mit Beteiligten im Engineering identifiziert werden. Die Austauschartefakte können ebenfalls im Rahmen von Dokumentenanalysen identifiziert werden. Bei der Kontextanalyse werden die Organisationseinheiten zwar als Black-Box betrachtet, falls jedoch detaillierte Prozessmodelle vorliegen, können diese auch innerhalb der Umfeld-

elemente analysiert werden, um relevante Schnittstellen zu identifizieren, z. B. In- oder Outputs des externen Prozesses. Die Beschreibung der Schnittstellen erfolgt mit den o. g. Sprachkonstrukten. Dabei ist darauf zu achten, dass im Einvernehmen mit den durch das Umfeldelement repräsentierten Stakeholdern und dem Engineering ein Zielzustand für den Artefaktaustauschfluss und die Austauschartefakte spezifiziert wird. Dies erfolgt typischerweise iterativ, da bspw. die IT-Systeme in beiden Organisationsbereichen angepasst werden können und sich so auch Änderungen hinsichtlich der verarbeitbaren Datenformate ergeben können.

Internes Prozessmodell

Ziel des internen Prozessmodells ist es, eine White-Box Sicht auf die Prozesse des Engineering-Systems zu erzeugen. Im Fokus stehen dabei die wesentlichen Aktivitäten und von ihnen genutzte Artefakte, die in einer sachlogischen Abfolge dargestellt werden. Sie ermöglichen eine erste Strukturierung und inhaltliche Beschreibung des Engineerings auf einem eher höheren Abstraktionsgrad. Das Prozessmodell dient daher als Grundlage zur Identifikation weiterer relevanter Elemente wie Methoden, Stakeholdern und technischer Ressourcen. Prozessmodelle sind zudem sehr geläufig und erleichtern so den Einstieg in die Strukturierung des Engineerings. Das interne Prozessmodell greift Konstrukte von OMEGA auf, nutzt als zugrundeliegende Modellierungssprache jedoch BPMN. Dadurch können Schwächen von OMEGA, wie eine fehlende explizite Verknüpfung von Modellelementen und die eingeschränkte Werkzeugunterstützung, eliminiert werden. Gleichzeitig können relevante Teile der Semantik von OMEGA genutzt werden und die Sprachkonstrukte mit den Sprachkonstrukten der weiteren Partialmodelle verknüpft werden. Die BPMN und UML werden dazu um neue Stereotypen erweitert, die relevanten Sprachkonstrukte sind in Tabelle 5-5 aufgeführt.

Innerhalb des Engineering-Systems werden die Prozesse des Engineerings beschrieben. **Swimlanes** erlauben es, das Engineering weiter zu untergliedern, um die Übersichtlichkeit zu verbessern. Sie können bspw. nach Domänen gegliedert sein, z. B. Projektmanagement, Systementwicklung, etc. Die Swimlanes können **Aktivitäten** enthalten, welche durch **Sequenzflüsse** in eine sachlogische Abfolge gegliedert werden können. Die Aktivitäten nutzen **Artefakte** als In- und Outputs, welche somit Informationsträger darstellen. Die in Prozessen identifizierten Artefakte weisen typischerweise einen eher hohen Abstraktionsgrad auf und umfassen oft nicht-atomare Informationsobjekte. Sie können über die **Datenassoziation** als In- oder Output mit einer Aktivität verknüpft werden. Die erwartete Durchgängigkeit beim Daten- bzw. Informationsaustausch kann analog zum Kontextmodell über die Datenassoziationen spezifiziert werden.

Das Modell kann durch Elemente ergänzt werden, welche die Analyse unterstützen. **Methoden** beschreiben ein regelbasiertes, planmäßiges Vorgehen zur Ausführung einer Aktivität und/oder Dokumentation ihrer Artefakte (vgl. Kap. 3.3.2). Sie werden immer mittels einer Assoziation mit einer oder mehreren Aktivitäten verknüpft. Methoden begründen

Tabelle 5-5: Übersicht über die für das interne Prozessmodell relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Swimlane *Lane (BPMN)*		Strukturiert die Prozesse innerhalb des Engineering-Systems, typischerweise in spezifischen domänespezifischen und -übergreifenden Tätigkeitsbereichen.
Aktivität *Activity (BPMN)*		Beschreibt eine Tätigkeit als Arbeistschritt eines Prozesses des Engineering-Systems.
Sequenzfluss *SequenceFlow (BPMN)*		Verbindet zwei Aktivitäten, um ihre sachlogische Abfolge zu beschreiben.
Artefakt «Artefact»		Spezifiert einen Informationsträger bzw. Arbeitsergebnis, welches als In- oder Output einer Prozessaktivität bzw. Methodenaufgabe zugeordnet werden kann.
Datenassoziation «DataLink»	«DataLink»	Ordnet ein Artefakt einer Aktivität zu. Je nach Richtung der Beziehung resultiert daraus ein In- oder Output für die Aktivität.
Methode «Method»		Definiert eine Methode, ein regelbasiertes, planmäßiges Vorgehen zur Ausführung einer Aktivität und/oder Dokumentation ihrer Artefakte, als Black-Box.
Technische Ressource «Ressource»		Beschreibt ein spezifisches, physisches Objekt, welches In- oder Outputs in Form von Artefakten mit Aktivitäten austauscht.
Lebenszyklusmgt.-Prozess «LCM-Process»		Beschreibt einen definierten Prozess, welcher ein Artefakt über dessen Lebenszyklus steuert, als Black-Box.
Assoziation *Association (UML)*		Verknüpft zwei Modellelemente durch eine gerichtete Beziehung. Artefakte oder Aktivitäten stellen immer das Ziel einer Assoziation dar.

aufgrund ihres operativen und definierenden Charakters eine Vielzahl von Anforderungen, die für die Engineering-IT relevant sind. Für die Ausführung einer Aktivität können zudem **technische Ressourcen** notwendig sein. Diese stellen physische Objekte dar, welche die Durchführung von Aktiväten unterstützen. Sie können Anforderungen an die Engineering-IT stellen, beispielsweise falls Schnittstellen zur automatisierten Datenübertragung an eine technischen Ressource gefordert sind. Die technischen Ressourcen können per Assoziation mit einer Aktivität verknüpft werden. Falls Artefakte über ihren Lebenszyklus verwaltet werden müssen, kann dies durch einen **Lebenszyklusmanagement-Prozess** (LZM-Prozess) modelliert werden, der mittels Assoziation einem Artefakt zugewiesen wird. Dieser ist ein definierter, als Black-Box betrachteter Prozess, wie z. B. ein Konfigurationsmanagementprozess. Der LZM-Prozess kann somit eindeutig benannt werden, was bereits die Identifikation inkonsistenter oder redundanter LZM-Prozesse ermöglicht. Bei genauerem Analysebedarf können die LZM-Prozesse durch einen separaten Prozessablauf detailliert werden.

Hinsichtlich des Vorgehens empfiehlt es sich, zunächst die Gliederung des Engineerings anhand der Swimlanes vorzunehmen. Diese können anschließend mit den Aktivitäten und Artefakten gefüllt werden. Abschließend erfolgt die Ergänzung der Methoden, technischen Ressourcen sowie der LZM-Prozesse. Die Modellierungstiefe der Prozesse sollte so gewählt werden, dass alle relevanten Methoden im Engineering erfasst werden. Zudem sollten alle wesentlichen Artefakte identifiziert werden, die Dokumente oder (Partial-)Modelle darstellen. Die weitere Zerlegung dieser Artefakte in atomare Elemente erfolgt im Rahmen des Methodenmodells.

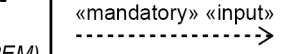
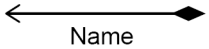
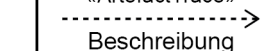
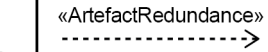
5.3.2.3 Methodenmodell

Die bereits im Prozessmodell identifizierten Methoden werden als White-Box ausdetailliert, indem für jede Methode ein Methodendiagramm erstellt wird. Dazu werden die wesentlichen Bestandteile der Methoden erfasst, um die im Prozessmodell gewonnen Informationen zu detaillieren. Während Prozesse einen eher höheren Abstraktionsgrad aufweisen und beschreiben, *was* erledigt werden muss, enthalten Methoden oft deutlich tiefergehende Informationen um zu beschreiben, *wie* Prozessschritte durchlaufen werden müssen oder woraus spezifische Artefakte bestehen. Ziel ist es dabei, die relevanten Artefakte und ihre Beziehungen sowie Anforderungen an die Engineering-IT systematisch abzuleiten. Dazu werden die in Tabelle 5-6 aufgeführten Konstrukte bereitgestellt.

Die Modellierung der Engineering Methoden basiert auf OMG SPEM, ergänzt um eine Modellierungsmethode. In einem Methodendiagramm werden die wesentlichen Tätigkeiten einer Methode als **Aufgabe** modelliert. Zwischen den Aufgaben ist im Gegensatz zum Prozessmodell keine formal modellierte Abfolge vorgesehen, da Bestandteile einer Methode oft situativ eingesetzt werden. Falls notwendig, können die Aufgaben durch **Methodenschritte** detailliert werden. Dadurch wird sichergestellt, dass keine relevanten Aktivitäten übersehen werden. **Artefakte** beschreiben in Prozessen Dokumente bzw. Modelle, welche aus verschiedenen Inhaltstypen und Attributen bestehen. Diese können mit Hilfe des Methodenmodells systematisch detailliert werden, da Methoden häufig auch beschreiben, wie Arbeitsergebnisse zu dokumentieren sind. Artefakte werden einer Aufgabe durch eine **In- bzw. Outputbeziehung** zugewiesen und können dabei auch als notwendig oder optional gekennzeichnet werden. Oftmals werden in Methoden auch **Attribute** und ihre Ausprägungen beschrieben. Sie beschreiben Eigenschaften eines Artefaktes und werden neben einem Namen durch einen Datentyp beschrieben. Durch Platzieren des Artefaktes aus dem Prozessmodell kann dieses durch untergeordnete, gleichartige Artefakte anhand von **Kompositionsbeziehungen** bis zur Ebene der Attribute detailliert werden.

Darüber hinaus beschreiben Methoden auch Beziehungen zwischen Artefakten, z. B. Beziehungen zur Sicherstellung der Traceability zwischen verschiedenen Arten von Entwicklungsartefakten. Diese in Methoden definierten Beziehungen werden durch sogenannte **Artefaktabhängigkeits-Beziehungen** modelliert. Darauf basierend können die Artefakte verschiedener Methoden z. B. in einer Artefaktbeziehungs-Matrix verglichen werden (vgl. Viewpoint PMV04b). Dazu werden die Namen der Objekte und ihre Steckbriefdaten ver-

Tabelle 5-6: Übersicht über die für das Methodenmodell relevanten Sprachkonstrukte und ihre Bedeutung

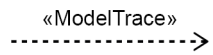
Bezeichnung	Notation	Beschreibung
Aufgabe *TaskDefinition (SPEM)*	Aufgabe durchführen	Beschreibt eine allgemeine Tätigkeit einer Methode, die im Rahmen ein oder mehrerer Prozessschritte zur Anwendung kommt und einige Stunden bis Tage dauert.
Methodenschritt *Step (SPEM)*	Schritt durchführen	Beschreibt einen Schritt, der zur Durchführung einer Aufgabe optional oder zwingend notwendig ist. Schritte gliedern Aufgaben und dauern max. wenige Stunden.
Artefakt «Artefact»	Name	Spezifiert einen Informationsträger bzw. Arbeitsergebnis, welches als In- oder Output einer Prozessaktivität bzw. Methodenaufgabe zugeordnet werden kann.
Attribut «ArtefactAttribute»	Name	Attribute spezifizieren die Eigenschaften eines Artefaktes. Sie werden durch einen Namen und einen Datentyp inkl. möglicher Auswahloptionen (bei Enumerationen) definiert.
Input-/Output-beziehung *ParameterType (SPEM)*	«mandatory» «input» 	Ordnet einer Aufgabe ein Artefakt zu. ParameterType (In- oder Output) sowie Optionality (Optional oder Notwendig) spezifizieren die Beziehung näher.
Komposition *Composition (UML)*	 Name	Beschreibt hierarchische Strukturen von Artefakten/Aufgaben, z.B. kann ein Spezifikationsdokument aus Anforderungen bestehen, welchen wiederum Attribute zugeordnet sind.
Artefakt-abhängigkeit «ArtefactTrace»	«ArtefactTrace»  Beschreibung	Spezifiziert durch die Beschreibung eine methodisch bedingte Abhängigkeit zwischen zwei Artefakten, z.B. notwendige Tracelinks.
Artefakt-redundanz «ArtefactRedundance»	«ArtefactRedundance» 	Markiert eine semantische Redundanz zwischen zwei Artefakten, die den gleichen Sachverhalt durch unterschiedliche methodische Artefakte beschreiben.
Artefakt-repräsentation «ArtefactRepresentation»	«ArtefactRepresentation»	Beschreibt die (gewollte) Nutzung einer Artefaktes bzw. seiner Attribute innerhalb eines übergeordneten Artefaktes in Form einer Referenz (ohne Redundanz).

glichen, um inhaltliche Überschneidungen zu identifizieren. Tritt der Fall auf, dass zwei Artefakte den gleichen Sachverhalt beschreiben, so werden diese in der Matrix durch eine **Artefaktredundanz-Beziehung** zwischen den beiden Artefakten gekennzeichnet. Somit können Überschneidungen von Artefakten identifiziert und dokumentiert werden. Es ist anzustreben, Redundanzen vollständig aufzulösen, indem das Artefakt der Kontrolle und Verantwortung einer einzigen Methode zugeordnet wird und stattdessen andere Methoden auf dieses zentral definierte Artefakt bzw. dessen Attribute zugreifen. Die Darstellung dieses erwünschten Zustandes erfolgt über **Artefaktrepräsentations-Beziehungen**, welche von dem untergeordneten (zu repräsentierenden) Artefakt zum übergeordneten Artefakt (in dem dieses dargestellt werden soll) gezogen werden. Wenn nur ausgewählte Attribute repräsentiert werden sollen, können diese über die Beziehung zugeordnet werden. Darüber hinaus können die nötigen CRUD-Operationen (Create = erzeugen; Read = lesen; Update = manupulieren bzw. ändern; Delete = löschen) für ein Artefakt oder Attribute im Modell als Attribut der Beziehung spezifiziert werden. Durch dieses Vorgehen können Redundanzen gezielt identifiziert und aufgelöst werden.

5.3.2.4 Anwendungsfälle

Während das Prozess- und das Methodenmodell stark die fachliche Perspektive fokussieren, beschreiben und definieren **Anwendungsfälle** eine durch ein IT-Werkzeug zu unterstützende, reproduzierbare Situation und die dabei vom IT-Werkzeug erwartete Unterstützung aus Anwendersicht. Sie erlauben so die Beschreibung typischer und oft wiederkehrender Aktionen in einem IT-Werkzeug, welche dann in verschiedensten Prozessen oder Methoden genutzt werden. Anwendungsfälle sind werkzeugunspezifisch. Sie erlauben somit eine zielgerichtete Unterstützung der Prozesse und Methoden sowie die systematische Evaluierung der Werkzeuge auf diese Fähigkeit hin. Die essentiellen Sprachkonstrukte sind in Tabelle 5-7 aufgelistet.

Tabelle 5-7: Übersicht über die für das Anwendungsfallmodell relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Anwendungsfall «UseCase»		Beschreibt eine allgemeine, häufig wiederkehrende, klar abgegrenzte Tätigkeit mit kurzem Zeithorizont, die in einem Softwarewerkzeug durchgeführt wird.
Rolle «Role»		Beschreibt eine temporäre, durch Aufgaben, Befugnisse und Verantwortungen abgrenzbare Funktion, welche durch eine Person ausgeübt werden kann.
Assoziation (ungerichtet) *Association (UML)*		Verknüpft einen Anwendungsfall mit einer Rolle durch eine ungerichtete Beziehung.
Modell-Trace «ModelTrace»		Stellt die Rückverfolgbarkeit zwischen zwei Elementen des Modells sicher und wird vom Ziel (z.B. Anwendungsfall) zur Quelle (z.B. Methodenschritt) erstellt.

Anwendungsfälle werden in SysML Use Case Diagrammen modelliert. Diese enthalten ein oder mehrere **Rollen**, die im Rahmen eines Anwendungsfalls mit einem IT-Werkzeug agieren, und die entsprechenden Anwendungsfälle. Rolle und Anwendungsfall sind durch eine **Assoziations-Beziehung** verbunden. Ein Anwendungsfall wird durch eine Ellipse dargestellt und durch eine Substantiv-Verb-Kombination beschrieben. Falls der Anwendungsfall aus einer dem Prozess- oder Methodenmodell zugehörigen Tätigkeit resultiert, kann diese über eine **Modell-Trace-Beziehung** zur Rückverfolgung mit dem Anwendungsfall verknüpft werden. Sollten Anwendungsfälle komplexere Stukturen annehmen, können diese über die Include-Beziehung der UML hierarchisiert werden.

Die Identifikation von Anwendungsfällen erfolgt typischerweise im Prozess- oder Methodenmodell. Dazu können Aktivitäten, Aufgaben und Schritte gezielt hinsichtlich einer gewünschten Werkzeugunterstützung analysiert werden. Neben einer textuellen Kurzbeschreibung werden die Vorbedingungen zum Durchlaufen des Anwendungsfalles, der dazu notwendige Auslöser und die anschließend vorliegenden Nachbedingungen spezifiziert. Ebenso werden die wesentlichen Punkte des Ablaufs lösungsneutral beschrieben. Dadurch

wird nicht nur ein einheitliches Verständnis sichergestellt, sondern es werden auch eindeutig abgegrenzte und reproduzierbare Testfälle geschaffen. Da die Anwendungsfälle häufige, in einem IT-Werkzeug ausgeführte Tätigkeiten definieren, können diese später im Rahmen von Tests genutzt werden, um IT-Werkzeuge hinsichtlich ihrer Gebrauchstauglichkeit zu bewerten. Daher ist bei der Formulierung und Beschreibung zu beachten, den Lösungsraum nicht unnötig einzuschränken, da unterschiedliche IT-Werkzeuge ggf. verschiedene Ansätze zur Unterstützung einer Aufgabe verfolgen und eine vergleichbare Bewertung ggf. nicht mehr möglich ist.

5.3.2.5 Funktionale Architektur

Basierend auf den Analysemodellen und den Anforderungen (vgl. Kap. 5.3.2.6) kann mit der funktionalen Architektur eine erste Konkretisierung der Lösung erfolgen. Die funktionale Architektur definiert durch die IT-Infrastruktur bereitzustellende Funktionen und damit eine lösungsneutrale Beschreibung. Dazu muss zunächst eine funktionale Struktur abgeleitet werden, welche die Funktionen hierarchisiert, die Funktionshierarchie. Aufbauend auf der funktionalen Struktur kann eine funktionale Architektur abgeleitet werden, welche die Funktionen miteinander in Beziehung setzt, um eine Umsetzung der Prozesse, Methoden und Anwendungsfälle zu ermöglichen. Die Sprachkonstrukte sind in Tabelle 5-8 dargestellt.

Tabelle 5-8: Übersicht über die für die Funktionshierarchie und die funktionale Architektur relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Funktionsgruppe «FunctionGroup»	«FunctionalGroup» Funktionen bereitstellen	Bündelt mehrere zusammenhängende Funktionen. Funktionsgruppen können untergeordnete Funktionsgruppen enthalten.
Funktion «Function»	«Function» Funktion bereitstellen	Beschreibt eine von einem IT-Werkzeug bereitzustellende Funktion. Diese muss genau einem IT-Werkzeug zuordbar sein.
Komposition *Composition (UML)*	Name	Beschreibt hierarchische Strukturen von Funktionen und Funktionsgruppen, bei der die untergeordneten Elemente von der Existenz eines übergeordneten abhängig sind.
Funktionsinstanz *Part (SysML)*	Funktionsinstanz bereitstellen	Repräsentiert eine spezifische Instanz einer Funktionsgruppe oder Funktion nach dem Prinzip von Block und Part der SysML.
Funktions-Schnittstelle «Fct-Interface»	«Fct-Interface» Funktionsinstanz bereitstellen	Definiert eine Schnittstelle einer Funktion oder Funktionsgruppe und beschreibt ihre Eigenschaften, indem sie einem Port zugeordnet wird.
Beziehung *Connector (UML)*	Beschreibung	Verbindet zwei Ports miteinander. Kann als Artefaktfluss (s.u.) oder Kontrollfluss (es werden Hilfsdaten, also keine definierten Artefakte ausgetauscht) definiert sein.
Artefaktfluss *ItemFlow (SysML)*	Objektname	Ordnet ein Artefakt einer Beziehung zu, d.h. ein Artefakt oder dessen Attribute werden zwischen zwei Ports ausgetauscht.

Funktionshierarchie

Die funktionale Struktur wird in einem Blockdefinitionsdiagramm der SysML in einem mehrstufigen Vorgehen modelliert. **Funktionsgruppen** bündeln stark zusammenhängende Funktionalitäten, welche eine hohe interne Komplexität und eine geringe äußere Komplexität aufweisen. Dies vereinfacht später die Zuordnung von Funktionen zu IT-Werkzeugen, da Schnittstellen zwischen verschiedenen Werkzeugen möglichst einfach gehalten werden können. Funktionsgruppen können eine Hierarchie bilden, die sich von einer Hauptfunktionsgruppe, welche alle von der betrachteten IT-Infrastruktur bereitzustellenden Funktionen umfasst, verzweigt. Zur Hierarchisierung werden **Kompositionsbeziehungen** genutzt. Es empfiehlt sich, ca. 7 ± 2 Elemente pro Hierarchieebene anzulegen [WRF+15, S. 8].

Als Grundlage für die Erstellung der funktionalen Struktur dienen vor allem das Prozess- und Methodenmodell sowie die Anforderungen. Weitere Modelle, wie die Anwendungsfälle oder Rollen und Personen, können mit einbezogen werden. Die Ableitung von Funktionsgruppen und Funktionen erfordert ein gewisses Maß an Fertigkeit vom Modellierenden, da sie nicht automatisiert werden kann [WLR+15, S. 199]. Im Rahmen dieser Arbeit sollen daher Heuristiken bereitgestellt werden, welche die Ableitung und Gruppierung von Funktionen erleichtern. Diese basieren auf den Heuristiken der FAS-Methode (vgl. Kap. 4.3.4) und sind für den vorliegenden Anwendungsfall entsprechend angepasst worden.

- *Ähnliche Prozesse und Methoden zusammenfassen:* Ähnliche Tätigkeiten aus Prozessen und Methoden können in Funktionsgruppen gebündelt werden, da diese oft ähnliche oder gleiche Ziele verfolgen. So können zum Teil Funktionsgruppen ganze Prozesse oder Methoden repräsentieren, da sie typischerweise eine Reihe von Aktivitäten zur Erreichung eines definierten Ziels beinhalten. Ein Beispiel wäre hier die Bündelung der Funktionen zum Konfigurationsmanagement.
- *Verarbeitung gleichartiger Artefakte:* Funktionen können anhand ihrer In- und Outputs gruppiert werden. Die verarbeiteten Artefakte sollten dazu möglichst gleichartig sein. Zum Abgleich können die Steckbriefe der Artefakte herangezogen werden, welche Aufschluss über die Art und Struktur geben. So können z. B. die Funktionen für die Handhabung textueller Anforderungen gebündelt werden und solche für grafische Architekturmodelle.
- *Bestehende Strukturen nutzen:* Insbesondere für den Fall, dass eine bestehendes IT-Infrastruktur erweitert oder angepasst werden soll, kann es sinnvoll sein, auf bereits bestehende funktionale Strukturen zurückzugreifen. Dabei muss jedoch bedacht werden, dass diese ggf. Einschränkungen aufweisen können, z. B. hinsichtich der Lösungsneutralität.

Die Funktionsgruppen werden in konkretere **Funktionen** zerlegt. Sie weisen eine Granularität auf, die es erlaubt, eine Funktion durch ein einzelnes IT-Werkzeug umzusetzen. Funktionen sind elementar für die Unterstützung von Prozessen sowie Methoden und stellen häufig K.O.-Kriterien für eine IT-Lösung dar, falls diese die Funktion nicht erfüllt.

Funktionen können teilweise aus Prozessaktivitäten, Methodenaufgaben, eher abstrakten Anwendungsfällen, Potentialen, Artefakten und technischen Ressourcen abgeleitet werden. Anwendungsfälle beschreiben im Gegensatz zu Funktionen eine reproduzierbar definierte Situation und die Nutzerinteraktion beim Ausführen einer Funktion. Zudem werden sie zur Bewertung der Nutzerinteraktion im Rahmen des Lösungsmodells (vgl. Kap. 5.3.2.7) den Funktionen zugeordnet. Eine weitere Hierarchisierung der Funktionen ist nicht vorgesehen.

Funktionale Architektur

Im Rahmen der funktionalen Architektur werden die Funktionen und Funktionsgruppen miteinander in Beziehung gesetzt. Dazu werden interne Blockdiagramme der SysML genutzt. Es bietet sich an, dem Grundsatz nach HABERFELLNER „vom Groben zum Detail“ zu folgen und die Funktionshierarchie top-down zu detaillieren. Dies bietet den Vorteil, dass bei Architekturentscheidungen die ganzheitlichen Auswirkungen besser abgeschätzt und überblickt werden können. Zudem wird dadurch ein arbeitsteiliges Vorgehen erleichtert, indem einzelne Funktionsgruppen an Verantwortliche zugewiesen werden können. Zunächst werden daher die Funktionsgruppen der höchsten Hierarchieebene aus der Funktionshierarchie im Diagramm platziert.

Es gilt nun, die **Beziehungen** zwischen den Funktionsgruppen zu modellieren. Nach den Grundlagen der Systemtheorie (vgl. Kap. 3.4.7) führen die Beziehungen der Elemente zur Schaffung von Emergenz. Im Rahmen dieser Arbeit dienen die Beziehungen dazu, ein emergentes Verhalten der IT-Infrastruktur zur Unterstützung der Methoden und Prozesse bereitzustellen. Demnach wird die Notwendigkeit von Beziehungen primär ausgehend von Prozess-, Methoden- und Anwendungsfallmodellen bestimmt. Zur Modellierung stehen Beziehungen zur Verfügung, die in Artefakt- und Kontrollflüsse aufgeteilt werden. Initial sollten die Artefaktflüsse modelliert werden. **Artefaktflüsse** repräsentieren den Austausch von Artefakten bzw. Attributen zwischen zwei Funktionsgruppen bzw. Funktionen. Dazu wird im Modell einer Beziehung ein Artefakt zugeordnet, was in einem Artefaktfluss resultiert. Ihre Notwendigkeit kann aus den Prozessen, Methoden und Anwendungsfällen resultieren. Zur Identifikation von Artefaktflüssen steht die Frage im Vordergrund: „Welche in Artefakten oder Attributen enthaltenen Informationen sind notwendig, um die Funktion mit den gegebenen Anforderungen zu realisieren“? Ausgangsbasis für die Beantwortung der Frage ist die genaue Analyse von Artefaktrepräsentationen zwischen verschiedenartigen Artefakten bzw. deren Attributen. Diese resultieren in der Regel in Artefaktflüssen.

Artefaktrepräsentations-Beziehungen sind als Quellen von Artefaktflüssen zu berücksichtigen. Ihre Spezifikation ist bei der Definition des Artefaktflusses zu beachten, da sie maßgeblich dessen Ausprägung bestimmmt. Zudem ist die Beziehung durch die geforderten CRUD-Operationen spezifiziert (s. Kap. 5.3.2.3). Die Prozesse und Methoden nehmen somit maßgeblichen Einfluss auf die Gestaltung der Architektur. So kann ein Lebenszyklusmanagement-Prozess z. B. eine Vielzahl von Artefaktbeziehungen erfordern, die in Abhängigkeit des Prozesses sehr unterschiedlich ausfallen können. Ist beispielsweise

die Erwartung, dass an einem zentralen Ort alle Änderungsanträge eines Änderungsmanagementprozesses und die damit verbundenen Artefakte aufgelistet werden können, so macht dies Artefaktflüsse zwischen allen Funktionen, die relevante Artefakttypen verwalten, und der Funktion für das Änderungsmanagement notwendig. Dies würde eine gänzlich andere Architektur ergeben, als wenn nur gefordert würde, dass alle Änderungsanträge zwar zentral verwaltet werden sollen, diese jedoch nur eine manuell einzugebende Objekt-ID der betroffenen Artefaktinstanzen enthalten.

Neben den Artefaktflüssen existieren **Kontrollflüsse**. Diese dienen dem Austausch von Hilfsdaten, die nicht unmittelbar Artefakte oder Attribute darstellen, sondern für die Umsetzung von Funktionen, Anforderungen oder Anwendungsfällen notwendig sind. Die ausgetauschten Daten werden über die Bezeichnung der Beziehung textuell spezifiziert. Artefaktabhängigkeits-Beziehungen zwischen andersartigen Artefakten führen häufig zu Kontrollflüssen, da über diese die methodisch vorzusehende Beziehung durch mindestens eine der betroffenen Funktionen umgesetzt werden muss. Kontrollflüsse können aber auch aus anderen Quellen resultieren.

Ergänzend zu den Beziehungen werden sogenannte Ports genutzt. Ein Port repräsentiert eine Schnittstelle einer Funktionsgruppe oder Funktion. Ports sind zwingend bei allen Beziehungen zu modellieren, die Grenzen von Funktionsgruppen überschreiten, um ihre modulare Nutzung zu ermöglichen. Bei Funktionen ist die Nutzung optional möglich. Ports werden über die Zuordnung von **Funktions-Schnittstellen** spezifiziert. Diese beschreiben über Attribute die bereitstellbaren und erwarteten Artefakte. Alternativ können die bereitstellbaren und erwarteten Kontrollflüsse textuell spezifiziert werden. Funktions-Schnittstellen können in einem separaten Viewpoint (vgl. *PFV04b – Funktionale Schnittstellen*) erstellt und anschließend wiederverwendbar den Ports in der Architektur zugeordnet werden.

Sind die Funktionsgruppen einer Hierarchieebene entsprechend miteinander in Beziehung gesetzt worden, werden die Sub-Elemente weiter ausdetailliert. Dazu werden die an den Ports der übergeordneten Funktionsgruppe ankommenden Beziehungen, wie auch in Bild 6-7 ersichtlich, an die inneren Elemente weitergeführt. Die Modellierung wird über die Funktionsgruppen über die Hierarchie so weit fortgesetzt, bis die Funktionsebene erreicht und mit den relevanten Beziehungen verbunden wurde.

5.3.2.6 Anforderungen

Die von der IT-Infrastruktur erwartete Funktionalität (funktionale Anforderung) und Eigenschaften (Qualitätsanforderung) werden über die Summe der Anforderungen beschrieben. Sie dienen daher in der Evaluierung als Bewertungskriterien. Sie können aus allen Partialmodellen heraus entstehen und werden somit ständig ergänzt. Die zur Anforderungsmodellierung relevanten Sprachkonstrukte sind in Tabelle 5-9 aufgeführt.

Tabelle 5-9: Übersicht über die für die Modellierung von Anforderungen relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Anforderung «EvaluationCriteria»	«EvaluationCriteria» Anforderungs-titel	Beschreibt eine textuelle Anforderung an die IT-Infrastruktur und dient der Bewertung der Lösung. Die Anforderung kann sich auf funktionale und Qualitätsaspekte beziehen.
Enthält-Beziehung *NestedClassifier (UML)*	⊕	Erlaubt die Hierarchisierung von Anforderungen und erlaubt es, diese genau einer Funktion zuzuordnen. Das übergeordnete Element befindet sich beim Kreis.
Modell-Trace «ModelTrace»	«ModelTrace» ----------→	Stellt die Rückverfolgbarkeit zwischen zwei Elementen des Modells sicher und wird vom Ziel (z.B. Anforderung) zur Quelle (z.B. Methodenschritt) erstellt.

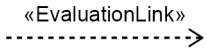
Die **Anforderungen** basieren auf dem von der SysML bereitgestellten Sprachelementen. Sie stellen Kriterien zur Bewertung der funktionalen bzw. qualitativen Eignung der IT-Infrastruktur oder ihrer Elemente dar. Daher werden sie immer Funktionen zugeordnet, was eine definierte Struktur ermöglicht. Dazu wird eine **Enthält-Beziehung** genutzt, das bedeutet, eine Anforderung ist exakt einer Funktion zugehörig. Innerhalb der Funktion darf nur eine Anforderung mit dem gleichen Titel existieren. Neben dem kurzen, prägnanten Titel wird jede Anforderung durch eine textuelle Beschreibung definiert. Zudem wird die Art der Anforderung festgelegt (funktionale Anforderung, Qualitätsanforderung, Rahmenbedingung; vgl. Kap. 4.2.1). Dabei ist zu bedenken, dass typischerweise zur Beschreibung nicht nur funktionale Anforderungen ausreichen. Qualitätsanforderungen sind ergänzend notwendig, um zu spezifizieren, in welcher Qualität die Funktionalität zu erbringen ist.

Anforderungen können häufig direkt oder indirekt aus Anwendungsfällen, Methodenschritten, Artefakten bzw. Attributen oder Potentialen abgeleitet werden. Teilweise basieren die Anforderungen auch auf Anwenderwissen, z. B. weil sich ein bestimmter Sachverhalt bewährt hat und auch durch die zu planende IT-Infrastruktur abbildbar sein muss. Solche Anforderungen sollen jedoch wohlüberlegt genutzt werden, da sie den Lösungsraum einschränken und ggf. „gewohnte“ Lösungen bevorzugen. Sie sollten nur genutzt werden, wenn die Nichterfüllung der Anforderung inakzeptabel ist. Zudem sollte bedacht werden, dass Anforderungen ausschließlich die funktionale Eignung beschreiben. Es sollten daher keine Funktionen bzw. Anforderungen hinsichtlich der Gebrauchstauglichkeit formuliert werden, um die Nutzenunabhängigkeit der Kriterien zu wahren (s. Kap. 5.3.2.7). Zur Sicherstellung der Rückverfolgbarkeit können Anforderungen anhand einer **Modell-Trace Beziehung** mit dem Modellelement, von dem die Anforderung abgeleitet wurde, verknüpft werden.

5.3.2.7 Lösungsmodell

Das Lösungsmodell beschreibt mögliche Kombinationen von IT-Werkzeugen als konsistente Lösungsansätze zur Umsetzung der funktionalen Architektur und bewertet diese. Dazu wird ein Lösungsraum aus geeigneten IT-Werkzeugen identifiziert und strukturiert. Aus diesen werden verschiedene, konsistente Lösungsvarianten gebildet. Dies geschieht im Wechselspiel mit der Definition der Komponentenarchitektur (vgl. Kap. 5.3.2.8). Die konsistenten Lösungsvarianten werden anschließend anhand der Anforderungen sowie der Anwendungsfälle und Schnittstellen bewertet. Das Lösungsmodell dient damit vorrangig der Identifikation und Auswahl einer geeigneten Lösung, bestehend aus IT-Werkzeugen. Dazu stehen die in Tabelle 5-10 dargestellten Sprachkonstrukte bereit.

Tabelle 5-10: Übersicht über die für die Modellierung des Lösungsmodells relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Softwaregruppe «SoftwareGroup»	«SoftwareGroup» Gruppen-name	Beschreibt eine Gruppe oder Art von IT-Werkzeugen, die potentiell als Suchfeld für die Zuordnung zu Funktionen geeignet ist.
Software-komponente «SoftwareComponent»	«SoftwareComponent» Komponenten-name	Repräsentiert ein verfügbares IT-Werkzeug, welches ein eigenständiges Werkzeug oder einen modularen Teil einer größeren IT-Lösung darstellt.
Assoziation *Association (UML)*		Verknüpft zwei Modellelemente durch eine gerichtete Beziehung.
Anforderung «EvaluationCriteria»	«EvaluationCriteria» Anforderungs-titel	Beschreibt eine textuelle Anforderung an die IT-Infrastruktur und dient der Bewertung der Lösung. Die Anforderung kann sich auf funktionale und Qualitätsaspekte beziehen.
Anwendungs-fall «UseCase»	**Anwen-dungsfall durchfüh-ren**	Beschreibt eine allgemeine, häufig wiederkehrende, klar abgegrenzte Tätigkeit mit kurzem Zeithorizont, die in einem Softwarewerkzeug durchgeführt wird.
Evaluierungs-beziehung «EvaluationLink»	«EvaluationLink» 	Ordnet ein IT-Werkzeug einem Bewertungskriterium oder einem Anwendungsfall zu und enthält die zugehörige, variantenspezifische Bewertung.
Variationspunkt *VariationPoint (Werkzeugspezifisch)*		Definiert die Gültigkeit eines Modellelements für eine Lösungsvariante. Wenn ein Element keinen Variationspunkt besitzt, ist es für alle Lösungsvarianten gültig.
Ergänzende Anforderung «AdditionalCriteria»	«AdditionalCriteria» Anforderungs-titel	Beschreibt eine textuelle Anforderung zur Bewertung der Lösung außerhalb der drei vorgebenen Bewertungsdimensionen.

Identifikation konsistenter Lösungsvarianten

Um den Lösungsraum gezielt aufzuspannen, werden **Softwaregruppen** definiert. Eine Softwaregruppe beschreibt eine Gruppierung von IT-Werkzeugen. Dazu wird in der Praxis meist nach Art des Werkzeugs unterschieden. Für die Funktion „Anforderungen handhaben“ ist es z. B. naheliegend, eine Softwaregruppe „Anforderungsmanagementwerkzeuge“

anzulegen. Dabei sind auch entsprechende Alternativen zu berücksichtigen. So sollte auch innerhalb der Softwaregruppe „PDM-Systeme“ gezielt gesucht werden, da auch diese teilweise entsprechende Funktionalitäten bereitstellen.

Ist der Lösungsraum durch die Softwaregruppen abgesteckt, so werden innerhalb dieser konkrete Softwarekomponenten identifiziert. Eine **Softwarekomponente** ist ein IT-Werkzeug, welches eine oder mehrere Funktionen der funktionalen Architektur umsetzen kann. Dabei muss dieses nicht zwangsweise ein eigenständiges Werkzeug darstellen, es kann auch ein modularer Teil einer größeren Softwarelösung sein, z. B. eines PDM-Systems oder Integrated-CASE-Systems. Bei der Identifikation von Softwarekomponenten ist darauf zu achten, dass für alle Funktionen mindestens eine Softwarekomponente verfügbar ist. Falls möglich, sollten die vielversprechendsten Alternativen zu einer Softwarekomponente ebenfalls mit aufgenommen werden. Die Zuordnung einer Softwarekomponente zu einer oder mehreren Softwaregruppen erfolgt durch eine **Assoziations-Beziehung**.

Ist ein geeigneter Lösungsraum identifiziert, gilt es, darin konsistente Lösungsvarianten herauszuarbeiten. Eine Lösungsvariante umfasst eine konsistente Kombination von Softwarekomponenten, welche zur Abdeckung aller Funktionen benötigt werden. Sie definiert somit sämtliche Elemente einer spezifischen Variante einer Komponentenarchitektur. Die Bildung einer Lösungsvariante erfolgt mithilfe eines morphologischen Kastens [Zwi67, S. 284ff]. Dieser spiegelt wider, welche Softwarekomponenten grundsätzlich zur Abdeckung einer Funktion geeignet sind (vgl. Viewpoint *PAV05a – Software-Funktions-Mapping*). Aus diesen wird dann eine spezifische Softwarekomponente mittels einer **Evaluierungsbeziehung** der Funktion zugeordnet. Da diese Zuordnung in der Regel nur für eine bestimmte Lösungsvariante gültig sein soll, wird die Evaluierungsbeziehung um einen **Variationspunkt** ergänzt. Dieser definiert zum einen, dass die spezielle Instanz der Evaluierungsbeziehung nicht für alle Varianten gilt. Zum anderen definiert der Variationspunkt die Gültigkeit zu einer spezifischen Lösungsvariante. Im Umkehrschluss ergibt sich, dass ein Modellelement ohne Variationspunkt für alle Lösungsvarianten gültig ist. Die Nutzung der Variationspunkte ermöglicht eine gezielte Nutzung der Variantenmanagementfunktionen eines Modellierungswerkzeuges. Damit können die verschiedenen Lösungsvarianten in einem Modell verwaltet werden. Je nach Bedarf erlaubt ein Modellierungswerkzeug dann die Filterung des Modellinhalts auf eine ausgewählte Lösungsvariante. Bild 5-9 zeigt, wie Softwarekomponenten über eine Allokationsmatrix auf Funktionen allokiert werden können.

Die Beurteilung der Konsistenz einer Lösungsvariante ist essentiell für ihre Umsetzbarkeit. Dabei ist insbesondere das Vorhandensein einer Schnittstelle zwischen den betroffenen IT-Werkzeugen, deren technische Kompatibilität und der notwendige Aufwand für die Kopplung zu beurteilen. Um den Aufwand in der praktischen Anwendung gering zu halten, sollte daher zunächst eine Abschätzung mittels verfügbarem Wissen und ggf. kurzer Recherche, z. B. anhand von Datenblättern, erfolgen. Darauf basierend kann meist abgeschätzt werden, ob es sinnvoll ist, eine bestimmte Kombination von IT-Werkzeugen für eine konsistente Lösungsvariante näher zu betrachten. Ist dies der Fall, sollte die

Allokations-Matrix V1 Fragestellung: „Welche Funktion (Zeilen) wird durch welche Softwarekomponente (Spalten) umgesetzt?“ Legende Anf.: Anforderung, Arch.: Architektur	SW-K1 (SK)	SW-K2.1 (SK)	SW-K2.2 (SK)	SW-K2.3 (SK)	SW-K3 (SK)
F Funktion 1			X		
F Funktion 2			X		
F Funktion 3		X			
F Funktion 4	X				

Allokations-Matrix V2 Fragestellung: „Welche Funktion (Zeilen) wird durch welche Softwarekomponente (Spalten) umgesetzt?“ Legende Anf.: Anforderung, Arch.: Architektur	SW-K1 (SK)	SW-K2.1 (SK)	SW-K2.2 (SK)	SW-K2.3 (SK)	SW-K3 (SK)
F Funktion 1			X		
F Funktion 2					X
F Funktion 3			X		
F Funktion 4				X	

Bild 5-9: Allokationsmatrix zur Zuordnung von Softwarekomponenten zu Funktionen für verschiedene Lösungsvarianten

Verifizierung der Konsistenz im Rahmen der Modellierung der Komponentenarchitektur erfolgen. Daher sollten diese nicht sequentiell, sondern parallel zum Lösungsmodell für die verschiedenen erfolgsversprechenden Lösungsvarianten entwickelt werden.

Evaluierung der Lösungsvarianten

Die Evaluierung der Lösungsvarianten stellt die ganzheitliche Bewertung der Lösungsvarianten hinsichtlich der verschiedenen Bewertungsdimensionen in den Vordergrund. Aus der Identifikation der konsistenten Lösungsvarianten ergibt sich eine Menge vielversprechender Lösungen. Diese müssen zunächst grob selektiert und eingegrenzt werden, da eine Evaluierung aller Lösungsvarianten mit einem hohen Aufwand einhergehen würde. Es empfiehlt sich, zwei bis fünf Varianten zu vergleichen, je nach Relevanz und Überschneidung.

Die Bewertung der Lösungsvarianten erfolgt auf Basis einer **Nutzwertanalyse** (NWA) nach ZANGEMEISTER [Zan14, S. 89ff]. Die NWA erlaubt die qualitative und mehrdimensionale Bewertung komplexer Handlungsalternativen zur Vorbereitung einer Entscheidung. Sie stellt in der Praxis oft das einzige sinnvolle Hilfsmittel unter diesen Bedingungen dar. Die prinzipiell ebenfalls qualitativen und mehrdimensionalen Verfahren der multi-attributiven Nutzentheorie und der Analytic Hierarchy Process (AHP) nach SAATY fallen aufgrund der strengen Kriterien für deren Einsatz bzw. des hohen Aufwands weg [Zan14, S. 7], [BL12, S. 173f]. Die NWA ist, trotz einiger Schwachstellen, ein einfach durchzuführender, systematischer und nachvollziehbarer Ansatz zur Entscheidungshilfe. Die NWA wird dazu vollständig durch das Lösungsmodell abgebildet. Hierzu werden mindestens die drei in Kap. 5.2.4 definierten Bewertungsdimensionen herangezogen, die jeweils durch einen, wie im Folgenden dargelegt, zu ermittelnden Nutzenwert N_{Ni} beschrieben werden.

Funktionale Eignung: Für die Bewertung der funktionalen Eignung einer Lösungsvariante i werden alle Funktionsgruppen, Funktionen und Anforderungen k hierarchisch („top-down“) wie in Tabelle 5-11 dargestellt gewichtet. Das Kriteriengewicht w_k wird direkt als Attribut des jeweiligen Elements gespeichert und ist damit unabhängig von der Lösungsvariante. Dabei empfiehlt es sich in Prozent zu werten, um eine direkte Bewertung zu vereinfachen. Die Summe der Kriteriengewichte einer Anzahl n bewerteter Kriterien in einer betrachteten Ebene müssen in diesem Fall 100 ergeben (s. Gleichung 5-1), weshalb

Tabelle 5-11: Ausschnitt aus einer Bewertung der funktionalen Eignung

Kriterium	IT-Werkzeug	Kriteriengewicht		UEQ Wert	Teilnutzwert	
		w'_k	w_k (norm.)		n'_{ik}	n_{ik} (norm.)
FG Funktionsgruppe		100%	100%		95%	95,000%
FG Unterfunktionsgruppe		50%	50%		74%	37,000%
F Funktion	SW-K1	25%	12,5%		65%	8,125%
UC Anwendungsfall 1	SW-K1	50%	6,25%	3	100%	6,250%
UC Anwendungsfall 2	SW-K1	50%	6,25%	-1,2	30%	1,875%
F ...	SW-K4	75%	37,5%		77%	28,875%

Legende FG Funktionsgruppe F Funktion UC Anwendungsfall norm.: normiert

als Einheit auch in Prozent gerechnet werden kann. Durch die mehrstufige Bewertung wird die Menge der gleichzeitig zu bewertenden Objekte reduziert. Die Kriteriengewichte sind über die verschiedenen Ebenen der NWA zu normieren. Die Ermittlung der Kriteriengewichte kann bei geringer Kriterienanzahl n und hoher Sachkenntnis direkt erfolgen. Da häufig jedoch eine hohe Anzahl von Kriterien vorliegt, empfiehlt sich auf den jeweiligen Bewertungsebenen ein paarweiser Vergleich auf Basis des AHP (vgl. Anhang A3.1).

$$\sum_{k=1}^{n} w_k = 100\%$$

Gleichung 5-1: Die Summe der Gewichtung von Kriterien einer Bewertungsebene muss immer 100 % ergeben

Alle **Anforderungen** einer Funktion werden auf der in Tabelle 5-12 dargestellten Ordinalskala hinsichtlich ihrer Erfüllung bewertet. Die Ordinalwerte müssen zur Anwendung im Rahmen der NWA mit einer Wertetabelle in Kardinalwerte transformiert werden. Tabelle 5-12 stellt dazu vorgeschlagene Werte bereit. Da die komplexen IT-Werkzeuge häufig anpassbar sind (vgl. Kap. 3.4.4), wird der Grad der notwendigen Anpassung berücksichtigt.

Die Kardinalwerte können in Abhängigkeit der betrachteten Organisation und ihrer Möglichkeiten zur Anpassung der IT-Werkzeuge justiert werden. So wird ein großer Konzern mit eigenem Bereich in der IT zur Implementierung tendenziell einen geringeren Wert für eine Erweiterung ansetzen als ein Kleinunternehmen, welches die Anpassungen extern beauftragen muss und hier keine Skaleneffekte aufweist. Der sich so ergebende Teilnutzenwert n_{ik} wird abschließend als Attribut der lösungsvariantenspezifischen Evaluierungsbeziehung gespeichert. Diese verbindet variantenspezifisch eine Anforderung mit einer Softwarekomponente. Zur Ermittlung des Nutzenwertes N_{Ni} der Bewertungsdimension ist die Gleichung 5-2 heranzuziehen. Die auf Funktions- bzw. Funktionsgruppenebene aggregierten Teilnutzenwerte werden normiert und können als variantenspezifisches Attribut an den jeweiligen Modellelementen eingetragen oder bei entsprechender Werkzeugunterstützung automatisch berechnet werden.

Tabelle 5-12: Abstufungen zur Bewertung der funktionalen Eignung mit den vorgeschlagenen Kardinalwerten

Bezeichnung	Definition	Kardinalwert
Kriterium erfüllt	Die betrachtete Anforderung wird im Standard der Software abgebildet (Out-Of-The-Box).	100%
Parametrisierung erforderlich	Die betrachtete Anforderung wird durch Parametrisierung (Beeinflussung des Systemverhaltens im Rahmen vordefinierter Funktionen, ohne Programmierung, z.B. über konfigurierbare Regelwerke) in der Software abgebildet.	90%
Erweiterung erforderlich	Die betrachtete Anforderung wird durch eine Erweiterung der Software umgesetzt. Die Software bleibt dadurch weiterhin ohne zusätzliche Anpassungen migrationsfähig.	60%
Customizing erforderlich	Die betrachtete Anforderung wird durch Anpassung / Erweiterung in der Software (Beeinflussung des Systemverhaltens mit Programmierung) abgebildet. Die Software ist dadurch nicht ohne zusätzliche Anpassungen migrationsfähig.	30%
Kriterium nicht erfüllbar	Die betrachtete Anforderung kann nicht oder nur mit unverhältnismäßigem Aufwand abgebildet werden.	0%

$$N_{Ni} = \sum_{k=1}^{K} n_{ik} \cdot w_k$$

Gleichung 5-2: Nutzenfunktion zur Bewertung einer Dimension für eine Lösungsvariante

Gebrauchstauglichkeit bzw. UX: Die Gebrauchstauglichkeit bzw. das UX wird ebenfalls, wie in Tabelle 5-13 dargestellt, im Rahmen einer NWA bewertet. Die bewerteten Kriterien k sind hier jedoch nicht die Anforderungen, sondern die **Anwendungsfälle**. Vorbereitend müssen die Anwendungsfälle daher den Funktionen zugeordnet werden, sodass auch hier durch die gleiche lösungsneutrale Struktur die Vergleichbarkeit der Lösungsvarianten sichergestellt ist. Die Anwendungsfälle werden analog zu den Anforderungen mit einem Kriteriengewicht w_k versehen. Die Bewertung erfolgt anhand des User Experience Questionnaire (UEQ) nach LAUGWITZ ET AL. [LHS08, S. 67ff]. Der Fragebogen ist effizient und einfach in der Anwendung (ca. 3-5 Minuten pro Fragebogen) und stellt die Stärken und Schwächen der betrachteten IT-Werkzeuge übersichtlich dar. Er ist darüber hinaus hinreichend validiert worden. Die Autoren stellen den Fragebogen und ein Auswertungswerkzeug auf Basis von Microsoft Excel frei unter [STH20a-ol] zur Verfügung. Es werden die sechs Skalen Effektivität, Durchschaubarkeit, Vorhersagbarkeit, Stimulation, Originalität und Attraktivität durch 26 bipolare Items auf einer siebenstufigen Skala (z. B. zwischen unverständlich und verständlich) gemessen [SHT14, S. 384ff], [Sar11, S. 194]. Der Fragebogen ist im Anhang A3.2 dargestellt. Die Fragebögen sind durch mehrere Probanden je Anwendungsfall und IT-Werkzeug auszufüllen. Dabei ist darauf zu achten, dass die Probanden möglichst einen Durchschnitt der späteren Anwender repräsentieren. Nach der Bewertung liegen die Bewertungsresultate x zwischen -3 und +3 vor, welche

ebenfalls hinsichtlich der Skala transformiert werden (-3 $\widehat{=}$ 0% und +3 $\widehat{=}$ 100%, dadurch ergibt sich eine Transformationsfunktion $n_{ik} = \frac{50}{3} \cdot x + 50$). Der gewichtete Gesamtwert aus allen sechs Skalen entspricht einem Teilnutzenwert n_{ik}, welcher analog der funktionalen Eignung als Attribut einer Evaluierungsbeziehung (zwischen Softwarekomponente und Anwendungsfall) gespeichert wird. Der Nutzwert hinsichtlich des UX kann dann ebenfalls durch Gleichung 5-2 ermittelt werden.

Tabelle 5-13: Ausschnitt aus einer Bewertung der User Experience

Kriterium	IT-Werk-zeug	Kriteriengewicht		UEQ Wert	Teilnutzwert	
		w'_k	w_k (norm.)		n'_{ik}	n_{ik} (norm.)
FG Systemkonzipierung unterstützen		100%	100%		95%	95,000%
FG Anforderungen handhaben		50%	50%		74%	37,000%
F Anf.-Objekte administrieren	SW-K1	25%	12,5%		65%	8,125%
UC Neue Rolle anlegen	SW-K1	50%	6,25%	3	100%	6,250%
UC Neuen Anf.-Objekt Typen anlegen	SW-K1	50%	6,25%	-1,2	30%	1,875%
F ...	SW-K4	75%	37,5%		77%	28,875%

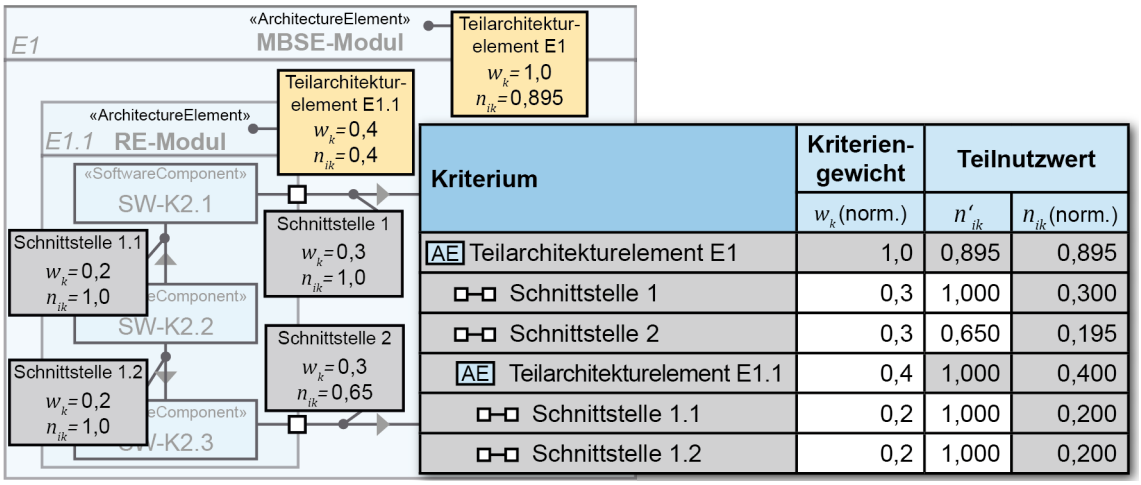
Legende FG Funktionsgruppe F Funktion UC Anwendungsfall norm.: normiert

Interoperabilität: Die Evaluierung der Interoperabilität erfolgt unter Betrachtung der Komponentenarchitektur, welche die Schnittstellen und Flüsse zwischen den beteiligten IT-Werkzeugen aufzeigt. Bild 5-10 stellt eine beispielhafte Bewertung dar. Interoperabilität wird hier als eine inhärente Eigenschaft von (Teil-)Architekturen betrachtet, welche aus der Bewertung ihrer Schnittstellen resultiert. Die Nutzung von Teilarchitekturen ist erforderlich, um bei komplexen Gesamtarchitekturen eine Hierarchisierung zu ermöglichen, damit die Menge der zu betrachtenden Objekte sinkt. Eine Teilarchitektur ist eine hierarchische Ebene der Komponentenarchitektur, die Schnittstellen k enthält. Ihre Interoperabilität wird durch den Teilnutzenwert n_{ik} ausgedrückt. Dieser resultiert aus den im Teilarchitekturelement enthaltenen Schnittstellen und ggf. weiteren enthaltenen Teilarchitekturelementen, welche eine Hierarchie bilden. Die Interoperabilität der Teilarchitekturelemente bzw. Schnittstellen wird durch den Teilnutzenwert n_{ik} ausgedrückt. Zunächst werden sowohl Teilarchitekturelemente als auch Schnittstellen gewichtet. Dabei gilt für die normierten Gewichte die Gleichung 5-3. Bei einer kleinen Anzahl n von Teilarchitekturen bzw. Schnittstellen werden diese durch einen direkte Bewertung mit einem Kriteriengewicht w_k versehen. Bei größeren Werten von n empfiehlt sich ein paarweiser Vergleich, basierend auf dem AHP (vgl. Anhang A3.1), zur Ermittlung der Kriteriengewichte. Dazu müssen die enthaltenen Softwarekomponenten und die durch sie umgesetzten Funktionen berücksichtig werden. Letztere können durch Modell-Trace Beziehungen zwischen den beiden Elementen nachvollzogen werden. Somit kann objektiviert werden, welche Relevanz eine Schnittstelle bzw. ein Teilarchitekturelement in der Gesamtbetrachtung besitzen sollte.

Nachfolgend wird die Ermittlung der Teilnutzenwerte n_{ik} für jede Teilarchitektur anhand ihrer Schnittstellen durchgeführt. Dazu wird jede Schnittstelle innerhalb der Teilarchitektur

$$\sum_{k=1}^{n} w_k = 1$$

Gleichung 5-3: Die Summe der Gewichtung von Kriterien bzgl. der Interoperabilität in einer Bewertungsebene muss immer 1 ergeben



Kriterium	Kriterien-gewicht	Teilnutzwert	
	w_k (norm.)	n'_{ik}	n_{ik} (norm.)
AE Teilarchitekturelement E1	1,0	0,895	0,895
Schnittstelle 1	0,3	1,000	0,300
Schnittstelle 2	0,3	0,650	0,195
AE Teilarchitekturelement E1.1	0,4	1,000	0,400
Schnittstelle 1.1	0,2	1,000	0,200
Schnittstelle 1.2	0,2	1,000	0,200

Bild 5-10: Beispielhafte Bewertung der Interoperabilität einer Architekturvariante

in Hinblick auf den von ihr geforderten Funktionsumfang, speziell bzgl. der geforderten Artefakt- oder Kontrollflüsse, bewertet. Zur Bewertung wird Tabelle 5-14 herangezogen. Der so transfomierte ordinale Teilnutzenwert n_{ik} einer Schnittstelle wird, wie auch w_k, als variantenabhängiges Attribut der Schnittstelle im Modell abgelegt. Für die beiden Werte von Teilarchitekturelementen gilt dies analog. Die Bewertung kann in der Komponentenarchitektur dargestellt werden, indem die Flüsse in Abhängigkeit ihrer Bewertung grün, gelb oder rot eingefärbt werden. Dadurch können Schwachstellen der einzelnen Lösungsvarianten hinsichtlich der Interoperabilität übersichtlich visualisiert werden. Der Gesamtnutzen aller Teilarchitekturen hinsichtlich der Interoperabilität N_{Ni} einer Lösungsvariante i ergibt sich nach Gleichung 5-2.

Die drei Werte N_{Ni}, jeweils für die funktionale Eignung, die Gebrauchstauglichkeit/UX und Interoperabilität, können anschließend für eine ganzheitliche Bewertung einer Lösungsvariante herangezogen werden. Da die Nutzwertanalyse eine vollständige oder zumindest bedingte Nutzenunabhängigkeit[6] der Kriterien voraussetzt, erfolgt im Rahmen der Arbeit ein mittelbarer Vergleich. Die Dimensionen der funktionalen Eignung und der Gebrauchstauglichkeit/UX sind (auch aufgrund ihrer gleichen Skalierung) unmittelbar vergleichbar. Da im Rahmen der funktionalen Eignung auch Funktionen bzw. Anforderungen enthalten sein können, die Schnittstellen erfordern und somit nicht nutzenunabhängig mit der Inter-

[6] Vollkommene Nutzenunabhängigkeit liegt vor, wenn zur Erreichung ein Zielkriterium nicht auch die Erreichung anderer Zielkriterien erfordert. Bedingte Nutzenunabhängigkeit liegt vor, wenn die Zielkriterien innerhalb festzulegender Intervalle unabhängig voneinander bewertbar sind [Zan14, S. 77f], [BL12, S. 152].

Tabelle 5-14: Abstufungen zur Bewertung der Interoperabilität mit den vorgeschlagenen Kardinalwerten

Bezeichnung	Definition	Kardinalwert
Voll erfüllt	Die betrachteten Anforderungen an die Schnittstelle werden durch die spezifizierte Lösung voll erfüllt.	1
Teilweise erfüllt	Die betrachteten Anforderungen an die Schnittstelle werden durch die spezifizierte Lösung nur teilweise erfüllt, jedoch ist die erwartete emergente Funktion in der Praxis bereits wahrnehmbar.	0,65
Unzureichend erfüllt	Die betrachteten Anforderungen an die Schnittstelle werden durch die spezifizierte Lösung nur so unzureichend erfüllt, dass die erwartete emergente Funktion kaum bzw. nicht wahrnehmbar ist.	0,35
Nicht erfüllt	Die betrachteten Anforderungen an die Schnittstelle werden durch die spezifizierte Lösung nicht erfüllt.	0

operabilität sind, ist ein unmittelbarer Vergleich der beiden resultierenden Nutzwerte von funktionaler Eignung und Interoperabilität nicht möglich. Um dies zu verhindern, wurde für die Interoperabilität bewusst eine andere Skalierung gewählt.

Falls weitere Dimensionen berücksichtigt werden müssen, erlaubt die Spezifikationstechnik die Nutzung von **ergänzenden Anforderungen**. Diese repräsentieren textuelle Anforderungen, die neben den o. g. drei Dimensionen zum Aufbau ergänzender Bewertungsdimensionen genutzt werden können. Sie müssen im Vergleich zu Anforderungen also nicht zwangsläufig einer Funktion zugeordnet werden, sich jedoch zu den vorgegebenen Bewertungsdimensionen nutzenunabhängig verhalten. Beispielsweise können so die Sicherheit, Zuverlässigkeit oder andere Aspekte berücksichtigt werden. Da die ergänzenden Anforderungen ebenfalls durch ein Gewicht und einen Teilnutzenwert beschrieben werden, können diese in die resultierende NWA analog zu den beschriebenen Dimensionen integriert werden.

Die vorteilhafteste Lösungsvariante weist den höchsten Gesamtnutzen aus funktionaler Eignung und Gebrauchstauglichkeit auf. Die Interoperabilität kann als aggregierter Wert eine Aussage über die Zielerreichung hinsichtlich der Durchgängigkeit geben. Werden zusätzliche Bewertungsdimensionen mit ergänzenden Kriterien genutzt, sind diese im Zielsystem entsprechend zu berücksichtigen. Weitere Faktoren (z. B. Kosten) können bei Bedarf ebenfalls mit in die Entscheidung einbezogen werden. Die detaillierten Ergebnisse der Analyse erlauben zudem die gezielte Betrachtung von Stärken und Schwächen der Lösungsvarianten und ggf. eine Verbesserung der Variante.

5.3.2.8 Komponentenarchitektur

Die Komponentenarchitektur beschreibt die logische Architektur einer spezifischen Lösungsvariante. Die im Lösungsmodell bereits zugeordneten und auf Konsistenz geprüften Softwarekomponenten (vgl. Kap. 5.3.2.7) werden dazu unter Berücksichtigung der funktionalen Architektur und der Anforderungen zueinander in Beziehung gesetzt. Die Komponen-

tenarchitektur stellt damit eine Applikationsarchitektur einer spezifischen Lösungsvariante dar, welche die grundlegenden beteiligten Softwarekomponenten und Schnittstellenkonzepte beschreibt. Sie kann daher auch als Prinziplösung aufgefasst werden, die grundlegende Softwarekomponenten definiert, jedoch nicht die detaillierte Spezifikation von konkreten, physischen Applikationen in Form von kompiliertem Code oder gar der Infrastruktur. Zur Modellierung stehen die in Tabelle 5-15 aufgeführten Sprachkonstrukte bereit.

Tabelle 5-15: Übersicht über die für die Komponentenarchitektur relevanten Sprachkonstrukte und ihre Bedeutung

Bezeichnung	Notation	Beschreibung
Teilarchitektur «ArchitectureElement»	«ArchitectureElement» Modulname	Bündelt mehrere zusammenhängende Softwarekomponenten zu einer gekapselten Einheit. Teilarchitekturen können untergeordnete Teilarchitekturen enthalten.
Softwarekomponente «SoftwareComponent»	«SoftwareComponent» Komponentenname	Repräsentiert ein verfügbares IT-Werkzeug, welches ein eigenständiges Werkzeug oder einen modularen Teil einer größeren IT-Lösung darstellt.
Komposition *Composition (UML)*	Name	Beschreibt hierarchische Strukturen von Softwarekomponenten und Teilarchitekturen, bei der die untergeordneten Elemente von der Existenz eines übergeordneten abhängig sind.
Komponenteninstanz *Part (SysML)*	Komponenteninstanz bereitstellen	Repräsentiert eine spezifische Instanz einer Softwarekomponente nach dem Prinzip von Block und Part der SysML.
Software-Schnittstelle «SW-Interface»	«SW-Interface» Komponenteninstanzname	Definiert eine Schnittstelle, die durch eine Teilarchitektur oder Softwarekomponente potentiell bereitgestellt werden kann, und beschreibt ihre Eigenschaften.
Schnittstellenfluss «SW-Connector»	Beschreibung	Verbindet zwei Ports miteinander und repräsentiert die tatsächlichen Eigenschaften der Schnittstelle und die darüber auszutauschenden Informationen.
Artefaktfluss *ItemFlow (SysML)*	Objektname	Ordnet ein Artefakt einer Beziehung zu, d.h. ein Artefakt oder dessen Attribute werden zwischen zwei Ports ausgetauscht.
Variationspunkt *VariationPoint (Werkzeugspezifisch)*	V1	Definiert die Gültigkeit eines Modellelements für eine Lösungsvariante. Wenn ein Element keinen Variationspunkt besitzt, ist es für alle Lösungsvarianten gültig.

Wie auch die funktionale Architektur wird die Komponentenarchitektur basierend auf einem internen Blockdiagramm der SysML abgebildet. Dazu werden die im Lösungsmodell für eine Lösungsvariante ausgewählten **Softwarekomponenten** im Diagramm platziert. Unter Berücksichtigung der allokierten Funktionen werden die Schnittstellen zwischen den Softwarekomponenten modelliert. Dabei ist zu beachten, welche Beziehungen die Funktion aufweist. Diese Beziehungen können auf die allokierte Softwarekomponente übertragen werden. Dabei müssen die Schnittstellen nicht eins zu eins den funktionalen Schnittstellen entsprechen. So kann eine funktionale Schnittstelle bspw. durch zwei Schnittstellen in der Komponentenarchitektur umgesetzt werden. Die Schnittstellen werden als **Schnittstellenfluss** modelliert und diesem zu übertragende Daten bzw. Informationen zugewiesen. Falls Artefakte bzw. Attribute übertragen werden, wird der Schnittstellenfluss um die Eigenschaften des **Artefaktflusses** erweitert. Daraufhin wird die entsprechende

Lösung ausgewählt und als Attribut des Schnittstellenflusses textuell spezifiziert. Die Festlegung einer Schnittstelle bei verschiedenen Möglichkeiten muss immer im Kontext der verbundenen Softwarekomponenten und der von ihnen bereitgestellten Schnittstellen betrachtet werden. Zudem wird die Schnittstelle hinsichtlich der Offenheit charakterisiert (neutral oder proprietär, vgl. Kap. 3.4.5). Wie in Kap. 5.3.2.7 beschrieben, wird für jede Schnittstelle auch die Erreichung der spezifizierten Interoperabilität modelliert, was in Bild 5-11 beispielhaft dargestellt ist.

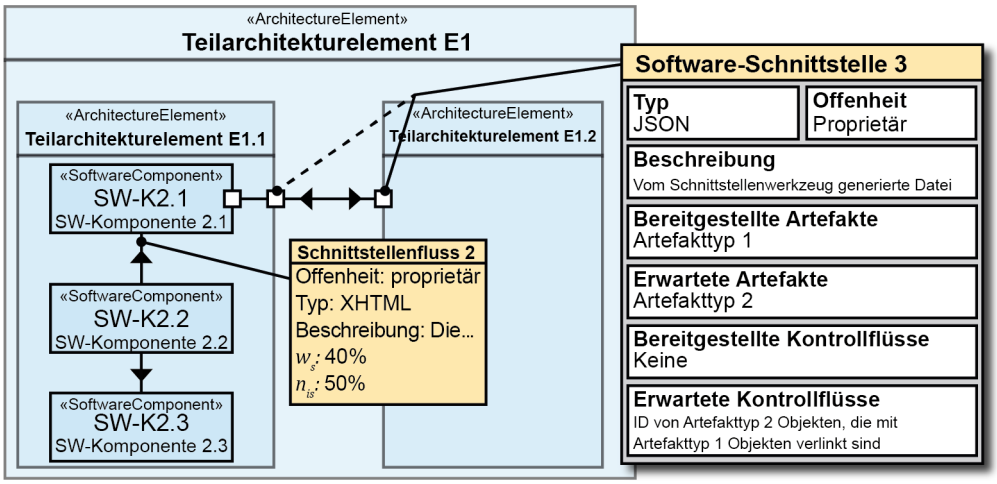


Bild 5-11: Nutzung des Profilmechanismus als Basis des entwickelten Sprachprofils

Diese Art der Modellierung erlaubt es, spezifische Eigenschaften der Softwarekomponenten und Architekturen und damit die Varianz der Lösungsvarianten zu berücksichtigen. Mit Hilfe der Ports kann darüber hinaus beschrieben werden, welche Schnittstellen potentiell durch eine Teilarchitektur oder eine Softwarekomponente angeboten werden können. Dies ist z. B. sinnvoll, wenn mehrere alternative Schnittstellen für die gleichen Informationen zur Verfügung stehen. Dazu kann eine **Software-Schnittstelle** beschreiben, welche Artefakte als In- oder Output verarbeitet und welche Schnittstellen prinzipiell bereitgestellt werden können. Ein Schnittstellenfluss definiert hingegen die für eine Lösungsvariante ausgewählten, spezifischen Eigenschaften. Es können die bereitstellbaren und erwarteten Artefakte zugeordnet bzw. Kontrollflüsse textuell spezifiziert werden. Zudem können die genutzten technischen Schnittstellen mit einer Beschreibung textuell definiert werden. Darüber hinaus kann festgelegt werden, ob die Schnittstelle offen oder proprietär ist. Sie werden vorzugsweise im Viewpoint *PAV04b – Software Schnittstellen* erstellt und spezifiziert. In der Komponentenarchitektur sind daher sowohl Software-Schnittstellen als auch die Schnittstellenflüsse zu modellieren.

Bei komplexen Architekturen sollte zudem mittels **Kompositionsbeziehungen** eine Hierarchie in einem SysML-Blockdefinitionsdiagramm erzeugt werden (vgl. Viewpoint *PAV03a – Architektur-Dekomposition*), um die Evaluierung im Rahmen des Lösungsmodells zu erleichtern und die Übersicht zu verbessern. Die so den Softwarekomponenten übergeordneten Architekturelemente werden als Teilarchitekturen bezeichnet. Eine **Teilarchitektur** bündelt mehrere zusammenhängende Softwarekomponenten zu einer gekapselten Ein-

heit. Wesentliches Kriterium zur Bildung von Teilarchitekturen sind dabei die internen Schnittstellen, die auch hier zu einer starken internen und möglichst geringen externen Komplexität führen sollen. Die Komplexität der Schnittstellen einer Teilarchitektur soll somit möglichst gering gehalten werden. Ein weiteres zu berücksichtigendes Merkmal ist die Anzahl der Schnittstellen.

5.4 Werkzeugunterstützung

Die Anwendung einer Modellierungssprache zum Aufbau eines Modells setzt eine Werkzeugunterstützung voraus. Dazu wird in Kapitel 5.4.1 ein Workshop-Kartenset vorgestellt. Zur Nutzung des Sprachkonzepts in einem Modellierungswerkzeug ist ein formalisiertes Sprachprofil notwendig, welches in Kapitel 5.4.2 definiert wird. Kapitel 5.4.3 stellt die Implementierung in einem kommerziellen Modellierungswerkzeug vor.

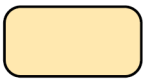
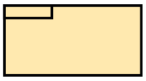
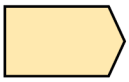
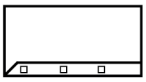
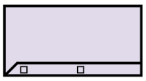
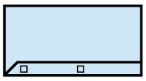
5.4.1 Kartenset zur Workshopmodellierung

An der Erfassung von Informationen zur Modellierung der Partialmodelle einer Engineering-IT-Architektur sind häufig eine Vielzahl von Stakeholdern beteiligt. Hier bieten sich insbesondere Workshops an, weil sie geeignet sind, Wissen einer Gruppe von Stakeholdern zu extrahieren und ein gemeinsames Verständnis zu schaffen [Poh08, S. 336ff]. Modelle können dabei Resultat oder Diskussionsgrundlage eines Workshops bilden [Kai14, S. 122f].

Das Kartenset soll die Modellierung in Workshops unterstützen. Dazu sind neun spezifische Workshopkarten definiert, die im PDF-Format zum Druck zur Verfügung stehen und in Tabelle 5-16 dargestellt sind. Damit die Anzahl der Karten begrenzt wird, müssen verwandte Variationen von Sprachelementen (z. B. Funktion und Funktionsgruppe) durch Ankreuzen eines entsprechenden Kästchens auf der Karte ausgewählt werden. Die Nutzung erfolgt auf Moderationspapier (Brown-Paper), sodass die Elemente durch die Karten repräsentiert werden und Beziehungen oder Flüsse mit einem Stift zwischen diesen erstellt werden. Sollen im Rahmen eines Workshops umfangreiche oder partialmodellübergreifende Beziehungen erfasst werden, bieten sich hierzu Matrizen auf Flipcharts oder Moderationspapier an.

Das für die Domänen in Tabelle 5-1 genutzte Farbschema wird durchgängig für die Notation der Sprachelemente genutzt (vgl. Anhang A1.1). Eine Ausnahme bilden die Anwendungsfälle, welche über die Partialmodelle den Analysemodellen zugeordnet sind. So werden die Objekte des Verhaltens & Tätigkeiten der Prozess- und Methodenmodelle sowie Anwendungsfälle in der Farbe *Orange* dargestellt. Artefakte und Attribute werden in *Weiß* repräsentiert. Die Funktionsgruppen und Funktionen werden in *Lila*, die Softwaregruppen und -komponenten sowie Teilarchitekturen werden *blau* notiert. Anforderungen sind grundsätzlich *rot*.

Tabelle 5-16: Moderationskartenset zur Anwendung des Kartensets in Workshops

Notation der Moderationskarte	Repräsentierte Sprachelemente	Benötigtes Material	Anmerkungen
	**Aktivität**	Workshopkarte 1	
	**Methode**	Workshopkarte 2	
	**Aufgabe**	Workshopkarte 3	
	Methodenschritt	Workshopkarte 4	
	**Artefakt, Attribut, Austauschartefakt**	Workshopkarte 5	Sprachelement durch Ankreuzen definieren
	Anwendungsfall	Moderationskarte oval, orange	
	**Funktionsgruppe, Funktion**	Workshopkarte 6	Sprachelement durch Ankreuzen definieren
	**Softwaregruppe, Softwarekomponente**	Workshopkarte 7	Sprachelement durch Ankreuzen definieren
	Anforderung	Workshopkarte 8	
	Rolle, Person, Potential, Dok. Objekt., Techn. Ressource, Strat. Ziel, LCM-Prozess	Workshopkarte 9	Sprachelement durch Ankreuzen definieren
	Variationspunkt	Klebepunkt, blau, 20mm	An variantenspezifisches Element /Diagramm kleben; Variantennummer per Flip-chartmarker ergänzen

5.4.2 Sprachprofil zur rechnergestützten Modellierung

Zur Darstellung und Analyse umfassender Modelle ist die Nutzung eines rechnergestützten Modellierungswerkzeugs notwendig. Zwar lassen sich Inhalte im Workshop papierbasiert gut erfassen und diskutieren, eine konsistente Dokumentation (insbesondere mit Varianten) und die Trennung zwischen Modell und Sicht sind aber kaum möglich. Vielmehr bedingt dies ein Werkzeug, welches ein Modellrepository und die darin enthaltenen formal abgebildeten Modellelemente verwalten kann.

In der Praxis haben sich, wie in Kap. 4.1 dargelegt, verschiedene Modellierungssprachen der OMG durchgesetzt. Diese weisen den Vorteil auf, dass sie von einer Vielzahl von Modellierungswerkzeugen unterstützt werden, über einen Profilmechanismus erweiterbar und auch untereinander verknüpfbar sind. Sie stellen bereits eine Vielzahl adaptierbarer Grundkonstrukte bereit, die jedoch nicht in vollem Umfang den Anforderungen des vorliegenden Anwendungsfalls genügen. Daher wird im Folgenden das im Rahmen dieser Arbeit

entwickelte STEITA Sprachprofil der **S**pezifikations**T**echnik zur Analyse, Gestaltung und Bewertung von **E**ngineering-**IT**-**A**rchitekturen (STEITA) dargestellt, welches, wie in Bild 5-12 ersichtlich, auf der SysML, BPMN und SPEM basiert und diese erweitert. SysML und SPEM basieren dabei auf der UML, welche auch die Grundlage für grundlegende Elemente des STEITA-Profils darstellt.

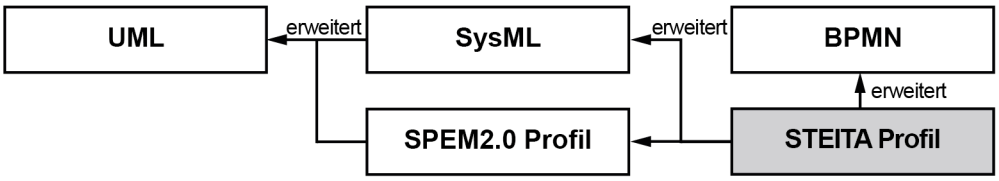


Bild 5-12: Nutzung des Profilmechanismus als Basis des entwickelten Sprachprofils

BPMN wird dabei vor allem als Grundlage für das Prozessmodell genutzt, da die Sprache die Darstellung sachlogischer Abläufe in Geschäftsprozessen fokussiert. Das SPEM-Profil wird als Grundlage für die Modellierung von Methoden genutzt. Die SysML wird vor allem als Basis für die Modellierung von Anforderungen und der funktionalen sowie der Komponentenarchitektur genutzt. Die UML wird für Anwendungsfälle und grundlegende Objekte genutzt. Das STEITA-Profil und die detaillierte Ableitung der Elemente sowie ihre Attribute sind in Anhang A1.3 dargestellt. Die Profilbeschreibung greift dazu die Metamodellelemente von UML, SysML, BPMN und SPEM auf, welche durch Rechtecke mit Strichlinie dargestellt werden. Der Profilierungsmechanismus erlaubt die Vererbung von Metaklassen mit Hilfe einer *Erweiterungsbeziehung*. Die Vererbung von Stereotypen erfolgt über eine *Generalisierungsbeziehung*. Daraus resultieren die Elemente des STEITA-Profils, welche durch Rechtecke mit Volllinie dargestellt werden. Die explizite Semantik der Profilelemente und die Zuordnung zu Partialmodellen ist in Kapitel 5.3.2 definiert.

Das STEITA-Profil kann prinzipiell mit Modellierungswerkzeugen umgesetzt werden, welche die Modellierung der UML und BPMN unterstützen sowie Sprachprofile der SysML und von SPEM implementiert haben. Im Rahmen der Umsetzung eines Softwarewerkzeugs zur rechnergestützten Modellierung im folgenden Kap. 5.4.3 wurde das Profil beispielhaft für ein kommerzielles Modellierungswerkzeug implementiert und ist im nachfolgend beschriebenen Plug-in enthalten.

5.4.3 Softwarewerkzeug zur rechnergestützten Modellierung

Für die praktische Anwendung der Spezifikationstechnik bietet die rechnergestützte Modellierung eine signifikante Unterstützung. Als Grundlage dafür soll das MBSE-Modellierungswerkzeug *Cameo Systems Modeler* des Herstellers *No Magic* dienen. Das Werkzeug unterstützt in der Grundkonfiguration UML und SysML. Zur Unterstützung der BPMN Funktionen ist das Cameo Business Modeler Plug-in erforderlich. Das SPEM-Profil kann über das Cameo SPEM Plug-in bereitgestellt werden. Im Folgenden soll das Werkzeug grob vorgestellt werden. Anschließend wird die Anpassung an das Sprachkonzept beschrieben. Für das Werkzeug wurde das in Kap. 5.4.2 vorgestellte Sprachprofil imple-

mentiert, um im Werkzeug entsprechende Sprachkonstrukte nutzen zu können. Weiterhin wurde die Notation entsprechend hinterlegt. Zudem wurden im Hinblick auf das in Kap. 5.3.1.3 vorgestellte Sichtenkonzept die Bedienoberfläche angepasst und entsprechende Diagrammtypen vordefiniert.

Die Bedienoberfläche des Werkzeugs ist in Bild 5-13 dargestellt. Der *Modellbrowser* zeigt die Modellelemente in den hierarchischen Strukturen des Modells an. In der *Diagrammfläche* werden ausgewählte Modellelemente mit ihrer definierten Notation als Sicht auf das Modell dargestellt. Das Modell kann durch Hinzufügen neuer Modellelemente, deren Typ in der *Diagrammpalette* gewählt werden kann, erstellt bzw. erweitert werden. Das *Attributfenster* zeigt die Attribute eines selektierten Modellelements an und erlaubt ihre Bearbeitung. Die weitere Steuerung des Werkzeugs erfolgt über die Menüleisten oder Kontextmenüs.

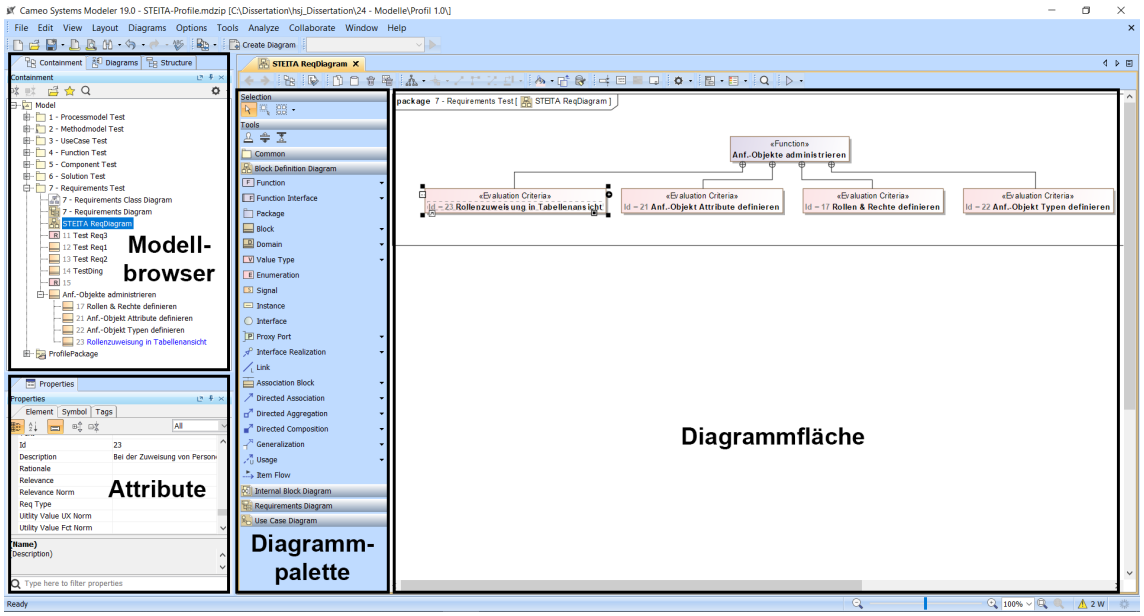


Bild 5-13: Benutzeroberfläche des Modellierungswerkzeugs Cameo Systems Modeler mit dem STEITA Plug-in

Die Implementierung verfolgt den Zweck, die Anwendung des Sprachkonzepts bei der praktischen Modellierung zu unterstützen. So wurde das Sprachprofil mit dem zugrundeliegenden Metamodell im Modellierungswerkzeug hinterlegt. Dazu wurde ein entsprechendes Profildiagramm in der UML-Notation erstellt. Somit ist es möglich, Modell-Elementen einen bestimmten Stereotyp des Profils zuzuweisen, welcher jeweils einem Metamodellelement entspricht. Dadurch kann das Modellrepository gemäß des dem STEITA-Profils zugrundeliegenden Metamodells aufgebaut werden. Zudem wurde für jeden Stereotyp des STEITA-Profils die graphische Notation angepasst, indem die passenden Farben zugewiesen wurden und ggf. Symbole hinterlegt wurden.

Darüber hinaus wurde das Werkzeug adaptiert, um insbesondere das Sichtenkonzept stärker zu unterstützen. Dazu wurden spezifische Diagrammtypen definiert. Dies ist u. a. auch notwendig, um die verschiedenen Sprachen wie UML und BPMN, z. B. im Rah-

men des Partialmodells Prozessmodell, nutzen zu können. Darüber hinaus wurden die Diagrammpaletten auf das STEITA-Profil angepasst. Somit stehen in Abhängigkeit vom gewählten Diagrammtyp die sinnvollerweise nutzbaren Sprachelemente zur Verwendung in der Diagrammfläche direkt zur Verfügung. Ein manuelles Zuweisen der Stereotypen zu einem Modellelement entfällt dadurch. Dies steigert die Gebrauchstauglichkeit der Implementierung erheblich.

Um interessierten Anwendern die Nutzung der Implementierung möglichst einfach zu erlauben, wurden die vorgenommenen Anpassungen bzw. Erweiterungen des Modellierungswerkzeugs in einem Plug-in exportiert. Dies ermöglicht die Nutzung der Anpassungen durch Import des Plug-ins in eine Standardinstallation des Modellierungswerkzeugs.

6 Anwendung und Bewertung

In diesem Kapitel wird die *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen* an einem Beispiel validiert. In **Kapitel 6.1** wird zunächst das Anwendungsbeispiel eingeführt. Es folgt die Anwendung, strukturiert nach den Phasen des Anwendungsmodells unter Nutzung des Sprachkonzeptes. Betrachtungsgegenstand ist dabei ein durchgängiges, aus der Landtechnik stammendes Anwendungsbeispiel, auf dessen Grundlage eine geeignete Engineering-IT-Architektur entwickelt und bewertet wird. Die Ergebnisse werden aus Geheimhaltungsgründen zum Teil vereinfacht dargestellt. Abschließend wird in **Kapitel 6.2** die Spezifikationstechnik anhand der in Kapitel 3.6 formulierten Anforderungen bewertet.

6.1 Anwendung der Spezifikationstechnik am Beispiel

Die Spezifikationstechnik wurde im Rahmen mehrerer Projekte in einem Landtechnikkonzern validiert. Die Validierung wurde über einen Zeitraum von ca. 3 Jahren durchgeführt. Sie kam dabei zur Analyse von Problemstellungen in verschiedenen Unternehmensbereichen zum Einsatz, die sich sowohl hinsichtlich der Organisationsgröße als auch der Art der zu entwickelnden Systeme (mechanische bzw. mechatronische Systeme sowie elektronische, softwarelastige Systeme) differenzieren.

6.1.1 Einführung des Anwendungsbeispiels

Die Ausgangssituation ist der im Landtechnikkonzern erkannte Bedarf, dass das Requirements Engineering des Unternehmens den geänderten Bedürfnissen durch die Entwicklung von ITS nicht mehr gerecht wird. Das Unternehmen entwickelte daher eine umfassende Methodik für das RE, welche sich an den Aktivitäten aus Kap. 4.2.1 orientiert und den im Unternehmen bestehenden Produktentstehungsprozess ergänzt. Ziel ist dabei, neben einer höheren Anforderungsqualität, die Rückverfolgbarkeit und lebenszyklusbegleitende Verwaltung der textuellen Anforderungen. Da diese im Zusammenspiel mit einer Architektur der ITS entwickelt werden, sollen auch Modelle im Sinne des MBSE mit Hilfe der SysML berücksichtigt werden können.

Zur Umsetzung sowohl des RE- als auch des Architekturprozesses werden geeignete IT-Werkzeuge benötigt. Ein Großteil der Anforderungen wird bislang in Microsoft Excel verwaltet, andere Organisationen des Unternehmens nutzen dazu ein dediziertes RE-Werkzeug. Weiterhin sind bereits Bestandteile einer integrativen PLM-Lösung im Einsatz, die bislang jedoch eher physisch-geometrische Aspekte der Produkte abbilden, z. B. CAD, CAE und CAM. Es gilt demnach, die bereits vorliegenden Prozesse und Methoden des RE zu analysieren, geeignete Anforderungen und IT-Architekturen abzuleiten und diese zu evalulieren.

6.1.2 Phase 1: Modellierung der Ausgangssituation

Kontextmodell

Zunächst wurden mit Hilfe des Kontextmodells die Organisationsstrukturen analysiert und daraus relevante Umfeldelemente abgeleitet. Ein solches stellt z. B. die Geschäftsführung dar. Für die weitere Analyse wurde die existierende Prozessdokumentation herangezogen und zudem ein Workshop mit Prozess- und Methodenverantwortlichen aus dem zentralen Entwicklungsmanagement durchgeführt. So konnten weitere Umfeldelemente wie Kunden oder Zulieferer identifiziert werden. Die Umfeldelemente wurden anschließend mit dem Engineering in Beziehung gesetzt. Die Workshopteilnehmer gaben dazu eine erste Einschätzung ab, in welche Richtung die Artefaktaustauschflüsse verlaufen. Dies ist in Bild 6-1 dargestellt. Anschließend wurden die über die Flüsse ausgetauschten Austauschartefakte identifiziert und spezifiziert. Dazu wurden erneut die Experten im Workshop befragt und die Prozessdokumentation genutzt, in der bereits eine Teilmenge der Artefakte aufgeführt war. Anhand des so entstandenen Diagramms konnte leicht die Vollständigkeit der Umfeldelemente und der Austauschartefakte geprüft werden.

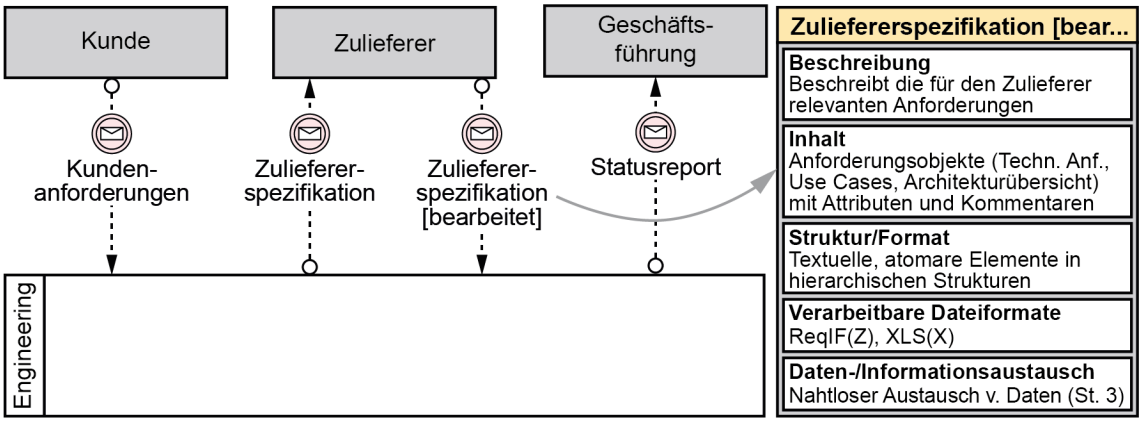


Bild 6-1: Ausschnitt des Kontextmodells mit steckbriefartiger Beschreibung eines ausgetauschten Artefakts

Strategische Ziele

Um einen Rahmen und übergeordnete Ziele für die folgenden Aktivitäten zu schaffen, wurden die strategischen Ziele hinter der Initiative identifiziert. Dazu wurde das Management des Unternehmensbereichs, der für die Definition der Prozesse und Methoden (basierend auf den Anwenderbedürfnissen) verantwortlich ist, sowie der Engineering-IT befragt. Ein wesentliches strategisches Ziel war die Harmonisierung der RE-Werkzeuge, um bestehende Werkzeuge möglichst abzulösen und die Daten in einem Werkzeug zu vereinen. Ein weiteres strategisches Ziel war die Schaffung einer bidirektionalen Traceability innerhalb der verschiedenen Anforderungsebenen und zu den Architekturmodellen. Als weiteres Ziel wurde die Einhaltung einer Few-Vendor-Strategie genannt, d.h. es sollen, wenn möglich, vom Unternehmen bevorzugte Anbieter von IT-Werkzeugen genutzt werden um die He-

terogenität der IT-Architektur zu reduzieren. Die Ziele wurden im Viewpoint *PRV01c – Strategische Ziele* erfasst und konnten so später bei Bedarf in den Partialmodellen genutzt werden.

Internes Prozessmodell

In den folgenden Schritten wurden die Prozesse innerhalb des Engineerings untersucht. Für die Gliederung der Swimlanes in den existierenden Prozessen wurden verschiedene Domänen identifiziert, u. a. Projektmanagement, Systementwicklung, Mechanik, Elektrik & Elektronik, Validierung und Produktionssystemplanung. Innerhalb der Swimlanes wurden die Prozesse zunächst anhand von existierenden Prozessabläufen, bestehend aus Aktivitäten und Artefakten, aufgebaut. In dem in Bild 6-2 dargestellten Beispiel ist die Konzeptionierung eines landtechnischen Systems dargestellt. Dies umfasst Prozessschritte von der Analyse der Kundenanforderungen aus Vertriebsgesellschaften über die Definition der Produktspezifikation durch das Produktmanagement hin zu funktionalen Konzepten für neu geforderte Produktfunktionalitäten. Die Anforderungen an das zu entwickelnde System werden aus technischer Sicht in der Maschinenanforderungsspezifikation definiert, auf deren Basis eine geeignete Architektur erstellt und die notwendigen Testfälle abgeleitet werden. Diese können im weiteren Entwicklungsverlauf in Verbindung mit digitalen Modellen des Systems und einem geeigneten Prüfsystem genutzt werden, um das Entwicklungsresultat automatisiert zu testen.

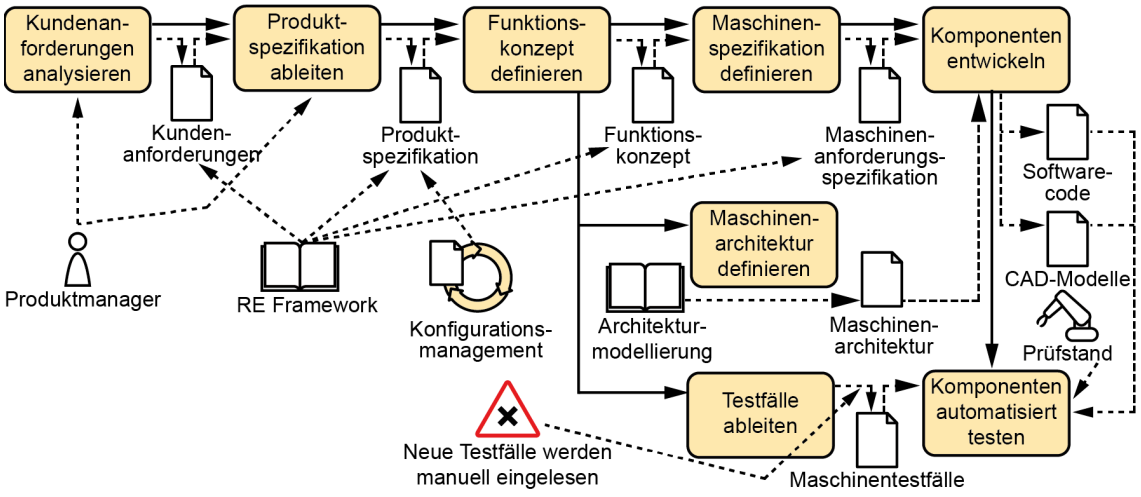


Bild 6-2: Ausschnitt von Modellelementen des internen Prozessmodells innerhalb der Swimlane Systementwicklung

Anschließend wurden die Prozesse analysiert, angepasst und in diesem Zuge um Methoden, technische Ressourcen und LZM-Prozesse erweitert. Dies erfolgte im Rahmen mehrerer Workshops mit den Prozess- und Methodenverantwortlichen. So wurden Methoden für das Requirements Engineering und die Architekturentwicklung identifiziert. Beim Testen der zu entwickelnden technischen Systeme wurde als Potential identifiziert, dass neue

Testfälle manuell an eine technische Ressource, den HiL-Prüfstand, übertragen werden müssen. Anschließend wurden alle LZM-Prozesse ermittelt und diese dann den Artefakten zugeordnet. Im Bild ist ersichtlich, dass die Anforderungsspezifikation und die Architektur beispielsweise einem Konfigurationsmanagement-Prozess unterliegen, welcher zeitliche und variantenspezifische Effektivitäten steuert. Zudem wurden strategische Ziele sowie Rollen, Personen und Potentiale identifiziert und mit den betroffenen Prozessenelementen verknüpft.

Methodenmodell

Die im Rahmen des Prozessmodells identifizierten Methoden wurden weiterführend in jeweils einem eigenen Methodendiagramm ausgestaltet. Dabei unterstützte der jeweilige Methodenverantwortliche, der in Workshops strukturiert befragt wurde. Die in Bild 6-3 dargestellte Methode „RE Framework“ besteht aus den sechs Phasen Identifizieren, Erheben, Dokumentieren, Analysieren, Reviewen und Verwalten. In Anlehnung an die SPEM Spezifikation werden diese als Aufgaben modelliert, da diese einen Arbeitszeitraum von einigen Stunden bis wenigen Tagen umfassen und in verschiedenen Prozessaktivitäten nutzbare Tätigkeiten beschreiben. Damit auch die darin enthaltenen Unteraktivitäten bekannt sind, wurden diese als Methodenschritte detailliert. Die Aufgabe „Dokumentieren“ kann beispielsweise in die Methodenschritte „Anforderungen textuell dokumentieren“ und „Anforderungen graphisch modellieren“ unterteilt werden. Dies erlaubte eine detailliertere und hierarchisch aufgebaute Analyse der Anforderungen, weil die Methoden systematisch untersucht werden. Als Quelle wurden zudem Methodendokumentationen wie der im Bild dargestellte und als Dokumentationsobjekt modellierte „RE Framework Reference Guide“ genutzt.

Die Artefakte und Attribute wurden vornehmlich auf Basis der Methodendokumentation ermittelt und hierarchisch strukturiert. So besteht ein Funktionskonzept aus Use Cases, Funktionen und technischen Anforderungen. Letztere können wiederum selbst Funktionen untergeordnet sein. Falls ein Use Case aus einem Kontextdiagramm resultiert, sieht die Methode vor, die beiden Objekte durch einen Use Case Tracelink miteinander zu verknüpfen. Dies wurde durch eine Artefaktabhängigkeits-Beziehung entsprechend dargestellt. Kontextdiagramme können, wie durch die Artefaktrepräsentations-Beziehung dargestellt, innerhalb eines Funktionskonzepts dargestellt werden. Das Bild zeigt den IST-Zustand, da Funktionen sowohl im Rahmen des RE Frameworks in Form textueller Beschreibungen als auch im Rahmen der Architekturentwicklungs-Methode in Form graphischer Repräsentationen vorkommen. Dieser Umstand wurde durch eine Artefaktredundanz-Beziehung kenntlich gemacht und sollte durch die Bereinigung einer der Funktionen und Nutzung einer Artefaktrepräsentation konsolidiert werden. Abschließend wurden strategische Ziele sowie Rollen, Personen und Potentiale identifiziert und mit den betroffenen Prozess- und Methodenelementen verknüpft.

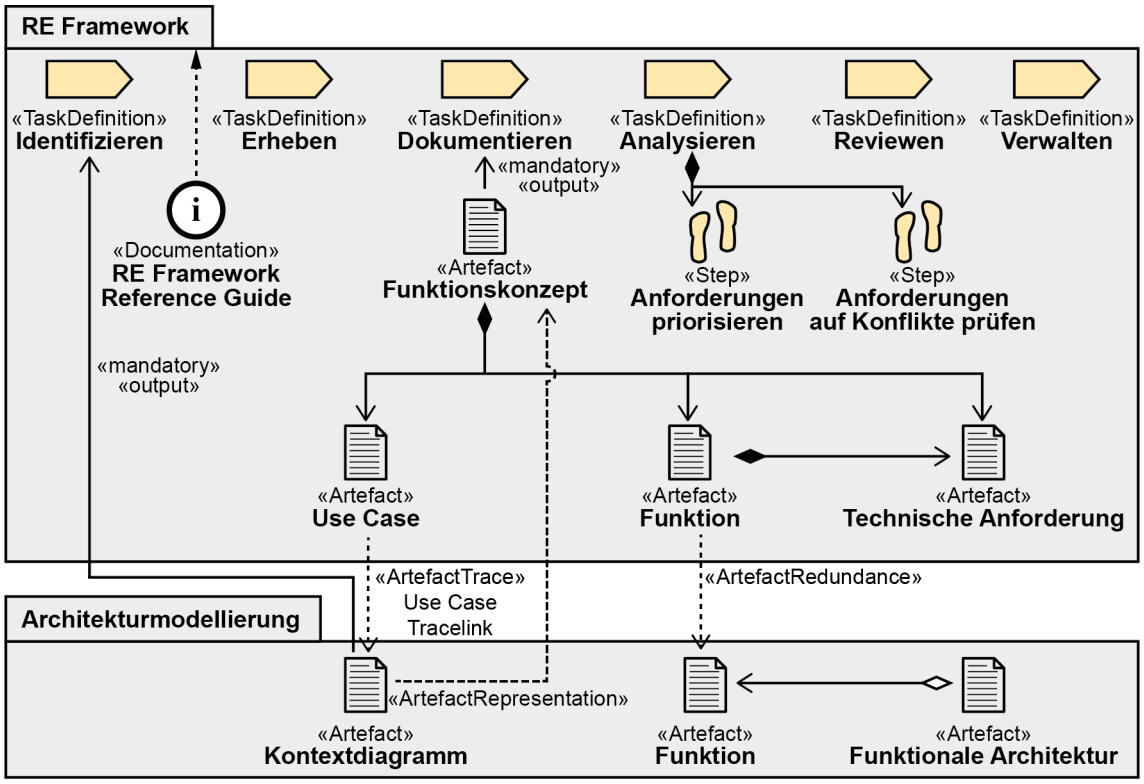


Bild 6-3: Ausschnitt aus den Methoden „RE Framework“ und „Architekturmodellierung“im Methodenmodell

6.1.3 Phase 2: Spezifikation des Zielbildes

Optimierung von Prozessen und Methoden

Die in den Prozess- und Methodenmodellen identifizierten Potentiale, Inkonsistenzen und Redundanzen wurden mit den betroffenen Prozess- bzw. Methodenverantwortlichen für die verschiedenen Prozesse und Methoden optimiert. Diese repräsentieren dann den SOLL-Zustand. Im zuvor gezeigten Beispiel des Funktionskonzeptes wurde beschlossen, dass für die Ausprägung des Artefakts der Funktionen ausschließlich die Domäne der Architekturmodellierung zuständig ist. Daher wurde die Redundanz der Funktionen im Modell entfernt, weil in Zukunft die Funktionen nur noch einmalig existieren sollen. Da jedoch weiterhin die Funktionen im Rahmen des Funktionskonzeptes genutzt werden sollen, wurde dies durch eine Artefaktrepräsentations-Beziehung verdeutlicht. Abschließend wurde kontrolliert, ob alle Potentiale im Modell berücksichtigt wurden und diese aus dem SOLL-Modell entfernt. Das optimierte Methodenmodell ist in Bild 6-4 dargestellt.

Anwendungsfälle

Basierend auf den festgelegten SOLL-Prozessen und Methoden wurden die Aktivitäten, Aufgaben und Schritte zusammen mit den Prozess- und Methodenverantwortlichen in Workshops mit Kleingruppen untersucht. Ziel war es, Anwendungsfälle abzuleiten. In Bild 6-5 kann z. B. aus dem Schritt „Anforderungen priorisieren“ der Bedarf für eine Werkzeug-

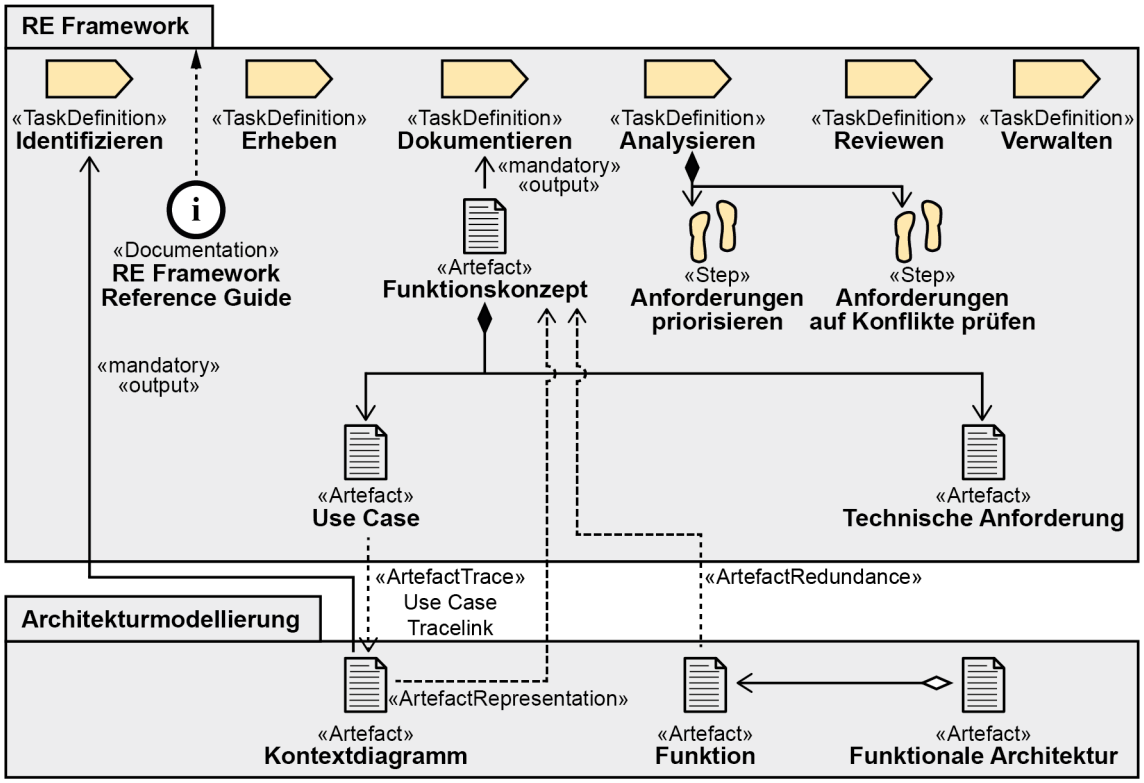


Bild 6-4: Ausschnitt aus den optimierten Methoden „RE Framework“ und „Architekturmodellierung“im Methodenmodell

unterstützung abgeleitet werden, um die Priorisierung von Anforderungen im IT-Werkzeug zu unterstützen. Jeder identifizierte Anwendungsfall wurde mit Rollen verknüpft und über einen Steckbrief näher beschrieben. Dadurch wird der Anwendungsfall reproduzierbar, da als Vorbedingung das Vorliegen von Anforderungen gemäß eines definierten Testdatensatzes für den Anwendungsfall definiert wurde. Der Anwendungsfall wurde erfolgreich abgeschlossen, wenn die Anforderungen priorisiert wurden und startet mit der ersten Priorisierung nach den im Ablauf vorgebenen Tätigkeiten. Im Bild ist ersichtlich, dass der Rolle zudem die Person „Max Mustermann“ zugeordnet wurde. Sie könnte bspw. später für einen Akzeptanztest eines implementierten IT-Werkzeugs herangezogen werden. Die Stakeholder wurden so Stück für Stück neben den Anwendungsfällen mit identifiziert. Die Viewpoints *PRV01a – Rollenidentifikation* und *PRV01b – Personenidentifikation* sorgten dabei für einen guten Überblick auch über verschiedene Themengebiete von Anwendungsfällen hinweg.

6.1.4 Phase 3: Definition des Zielsystems

Funktionshierarchie

Zur Identifikation der Funktionen wurden in einem Workshop mit den Prozess- und Methodenverantwortlichen gemeinsam zunächst die Prozess- und Methodenmodelle und anschließend die Anwendungsfälle mit Hilfe der Heuristiken analysiert. Identifizierte

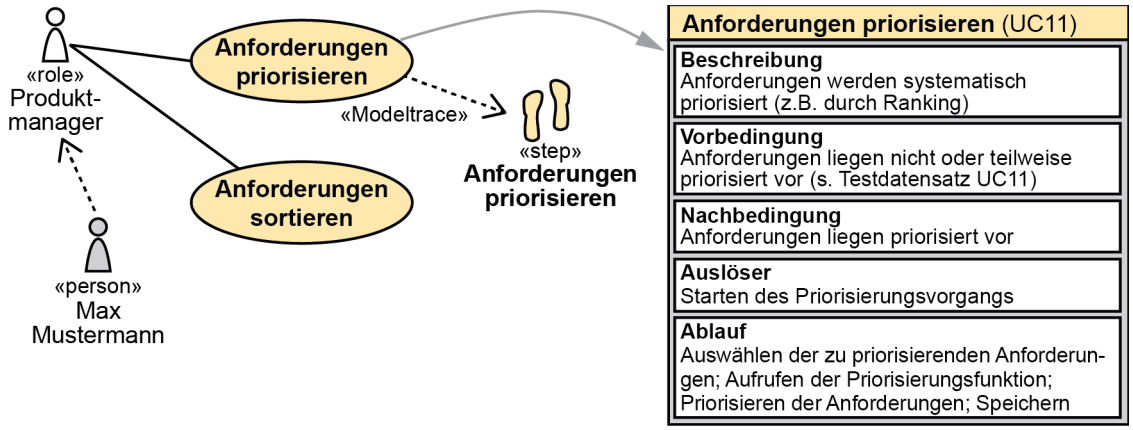


Bild 6-5: Ausschnitt aus den Anwendungsfällen

Elemente wurden zunächst auf einer Pinnwand mit den Workshopkarten dokumentiert und in Form eines Baumes strukturiert. Noch nicht berücksichtigte Funktionen oder fehlende Funktionsgruppen können durch eine erneute Kontrolle der so entstandenen Funktionshierarchie gegenüber den o. g. Modellen auf Vollständigkeit geprüft und ergänzt werden. Durch die abnehmenden Abstraktionsgrade wurde die Analyse der Funktionen vom Groben zum Detail unterstützt. Die Klassifizierung der Elemente als Funktionsgruppe oder Funktion wurde erst nach der Hierarchisierung durch die Workshopteilnehmer durch Ankreuzen des entsprechenden Feldes auf der Workshopkarte vorgenommen. Das Modell wurde für die weitere Verwendung in ein IT-werkzeuggestütztes Modell überführt und anschließend gereviewt.

Bild 6-6 stellt einen Ausschnitt dar und zeigt eine Funktionsgruppe „Anforderungen handhaben“ sowie eine weitere Funktionsgruppe „Architekturmodelle handhaben“. Dazu wurde die zweite Heuristik angewandt, da die Analyse der Artefakte ergab, dass es zwar unterschiedliche Anforderungsobjekte gibt, diese aber alle textuell spezifiziert werden. Gleiches gilt für die verschiedenen Architekturobjekte, z. B. Funktionen als Teil einer funktionalen Architektur. Da sowohl Anforderungen als auch Architekturen dem Konfigurationsmanagement unterliegen, kann über die erste Heuristik eine weitere Funktionsgruppe „Konfigurationen steuern“ abgeleitet werden, da hier gleiche Prozesse unterstützt werden sollen.

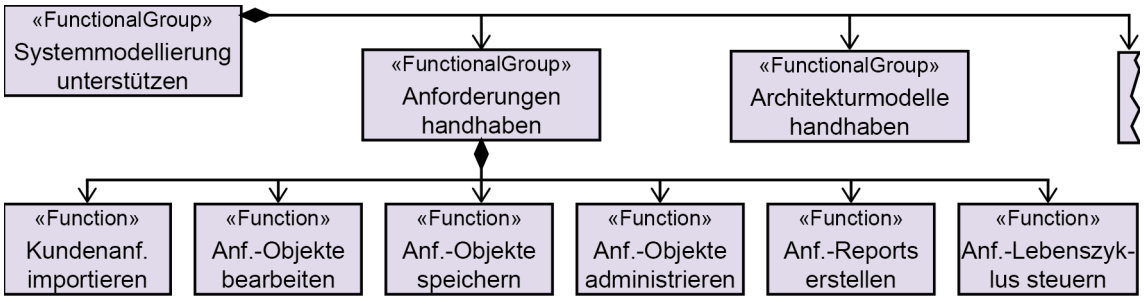


Bild 6-6: Ausschnitt aus der Funktionsstruktur

Die Funktion „Kundenanforderungen importieren“ resultiert aus der Prozessaktivität „Kundenanforderungen analysieren“ bzw. dem daraus resultierenden Artefakt „Kundenanforderungen“. Diese werden im vorliegenden Unternehmen typischerweise als PDF-Dokument übergeben, sodass sich dieses Artefakt in Form und Struktur von den anderen Anforderungsartefakten unterscheidet und ein automatisiertes Einlesen der Kundenanforderungen erforderlich macht. Ergänzend waren auch Hilfsfunktionen notwendig, die nicht direkt aus den Tätigkeiten in Prozessen und Methoden abgeleitet werden konnten. So war es beispielsweise vielfach notwendig, dass Objekte bearbeitet bzw. gespeichert oder Einstellungsparameter administriert werden sollen, z. B. für die Anforderungen und die Architekturelemente. So können für jede Funktionsgruppe die entsprechenden Funktionen ermittelt werden.

Funktionale Architektur

Die Funktionen wurden im folgenden Schritt miteinander in Beziehung gesetzt. Bild 6-7 zeigt einen Ausschnitt der resultierenden funktionalen Architektur. Zu deren Erstellung wurden Workshops mit den Prozess- bzw. Methodenverantwortlichen durchgeführt. Damit die Workshops in einem kleineren Teilnehmerkreis durchführbar waren, wurde mit ausgewählten Funktionsgruppen gestartet und die Beziehungen zwischen ihnen analysiert. So konnten die Architektur schrittweise aufgebaut und die Beziehungen aus verschiedenen Perspektiven vervollständigt werden. Als Grundlage zur systematischen Arbeit wurden die Beziehungen der Artefakte im Methodenmodell untersucht.

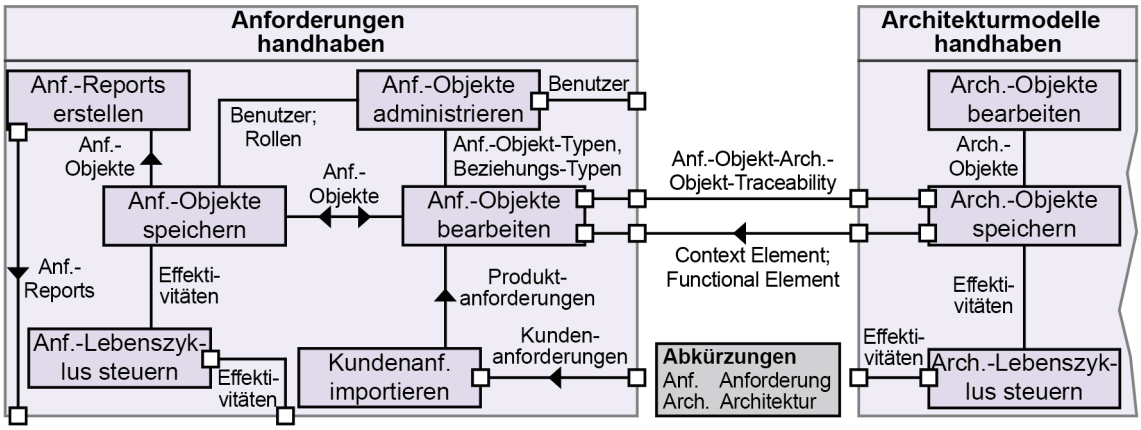


Bild 6-7: Ausschnitt aus der funktionalen Architektur

Betrachtet man die Funktionsgruppen „Anforderungen handhaben“ und „Architekturmodelle handhaben“, so konnten im Methodenmodell schnell erste notwendige Artefaktflüsse identifiziert werden. Da die Funktion „Anforderungen handhaben“ die textuellen Anforderungen fokussiert und „Architekturmodelle handhaben“ die visuellen Architekturelemente, ist aufgrund der Artefaktrepräsentations-Beziehungen zwischen dem Funktionskonzept und dem Kontextdiagramm in Bild 6-3 ein Artefaktfluss zwischen den beiden Funktionen notwendig. Hier wurde festgelegt, dass lediglich eine graphische Repräsentation des Kon-

textmodells im Funktionskonzept eingebettet werden soll. Daher reicht in diesem Fall eine unidirektionale Beziehung aus, welcher das Artefakt „Context Element“ zugeordnet ist.

Da zwischen Anforderungs- und Architekturobjekten im Methodenmodell eine Artefaktabhängigkeitsbeziehung zur Sicherstellung der Rückverfolgbarkeit zwischen den Objekten vorliegt, liegt hier die Notwendigkeit für einen Kontrollfluss nahe. Dieser wird im Beispiel als „Anf.-Objekt-Arch.-Objekt-Traceability“ bezeichnet. Weitere Kontrollflüsse sind für die Benutzerauthentifizierung notwendig. So müssen Benutzerdaten aus dem Identity-Management für die beiden o. g. Funktionen zur Verfügung gestellt werden. Dadurch kann eine Anforderung erfüllt werden, welche fordert, dass eine Anmeldung für die Nutzung der beiden Funktionen mit dem Windows-Benutzernamen und Kennwort möglich sein soll.

Anforderungen

Im Rahmen der bereits durchgeführten Workshops oder Interviews kamen stellenweise Anforderungen auf, die direkt auf jeweils einer Workshopkarte festgehalten wurden. Durch Prozess- und Methodenverantwortliche, aber auch durch Anwender wurden den Funktionen bereits erhobene Anforderungen zugeordnet. Zur Strukturierung der weiteren Anforderungserhebung wurde jeweils eine Funktion ausgewählt, für welche dann anschließend die Anforderungen in den Prozess- und Methodenmodellen sowie Anwendungsfällen erhoben wurden. Dabei galt es auch, Potentiale sowie Stärken bestehender Lösungen zu berücksichtigen. Die Workshops bzw. Interviews konnten so mit wenigen Teilnehmern durchgeführt werden, da nur die Experten für die jeweilige Funktion anwesend sein mussten. Für die Dokumentation der weiteren Anforderungen mit den Experten empfiehlt sich, zur Reduzierung des Aufwandes und zur Vermeidung von Übertragungsfehlern, die direkte Arbeit im rechnerbasierten Modell.

In dem in Bild 6-8 dargestellten Ausschnitt wurde die Funktion „Anf.-Objekte administrieren“ hinsichtlich der zugehörigen Anforderungen analysiert. Da sich die Funktion auf Anforderungsobjekte bezieht, wurden Prozesse, Methoden und Anwendungsfälle mit Bezug zu Anforderungen analysiert. So war schnell ersichtlich, dass verschiedene Rollen mit den Anforderungen arbeiten. Da diese jedoch in unterschiedlichem Maße im Prozess bzw. der Methode mitwirken, müssen deren Rechte rollenspezifisch eingeschränkt werden. So leitet sich die Anforderung ab, dass das Werkzeug das Anlegen verschiedener Rollen und Nutzer erlaubt. Da bekannt ist, dass die Vergabe der Rollen aufwändig sein kann, wird von den betroffenen Administratoren gefordert, dass Rollen einem Nutzer tabellarisch zuordbar sein müssen. Da im Methodenmodell ersichtlich ist, dass es verschiedene Arten von Anforderungen gibt, muss das IT-Werkzeug die Administration verschiedener Anforderungstypen erlauben. Jede ermittelte Anforderung wurde, wie im Bild dargestellt, unmittelbar mit dem entsprechenden Titel, der Beschreibung und dem Status versehen. So war sichergestellt, dass die Bedeutung der Anforderung sich später nicht mehr durch Interpretation oder einen anderen Kontext veränderte.

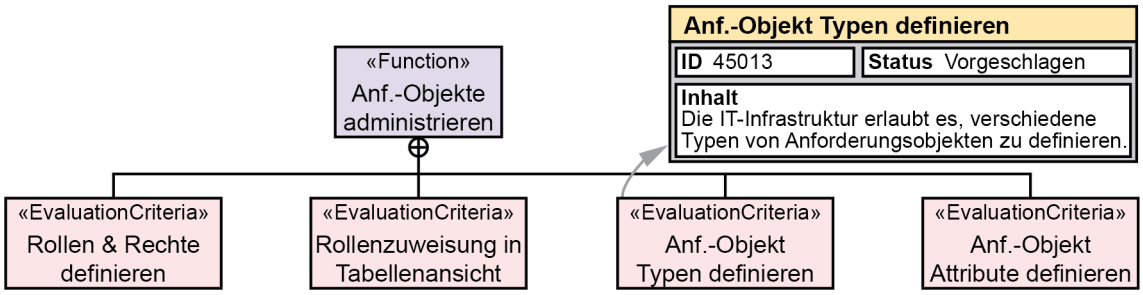


Bild 6-8: Ausschnitt von Anforderungen, die einer Funktion untergeordnet sind

6.1.5 Phase 4: Definition und Bewertung von Lösungsvarianten

Identifikation konsistenter Lösungsvarianten

Um geeignete Lösungsvarianten zu identifizieren, wurden zunächst die Anforderungen und Anwendungsfälle den jeweils passenden Funktionen zugeordnet. Zudem wurden die Elemente des so entstandenen Zielsystems gewichtet, was im Rahmen eines Workshops mit den Methoden-, Prozess- und IT-Verantwortlichen durchgeführt wurde. Dabei wurden nicht nur die Anforderungen für die funktionale Eignung, sondern auch die Anwendungsfälle für die Bewertung der Gebrauchstauglichkeit gewichtet.

Anschließend wurden durch IT-Verantwortliche geeignete Softwaregruppen, basierend auf der Analyse der Funktionshierarchie, festgelegt. Dabei wurden die Softwaregruppen „Anforderungsmanagementwerkzeuge“ sowie „Architekturmodellierungswerkzeuge“ definiert. Da auch einige PDM-Systeme Funktionen aus diesen Bereichen bereitstellen können, wurde eine gleichlautende Softwaregruppe ergänzt. Darin wurden gezielt geeignete Softwarekomponenten identifiziert, welche mindestens eine der relevanten Funktionen erfüllen konnten. Die Dokumentation erfolgte anhand von Workshopkarten am Brown-Paper. Durch Linien wurden die Softwarekomponenten den Softwaregruppen zugeordnet. Aufgrund des vorhandenen Marktwissens konnte eine Vielzahl der Funktionen allokiert werden. Wegen der Komplexität, die sich durch die Varianten ergibt, wurde die Morphologie direkt im rechnerbasierten Modell abgebildet. So wurde auf Ebene einzelner Funktionsgruppen und der darin enthaltenen Funktionen diese zu vielversprechenden Softwarekomponenten zugeordnet. Hier war eine gute Marktkenntnis sowohl durch die Prozess- und Methoden- als auch die IT-Verantwortlichen sehr hilfreich. Fehlt diese, so sollte unbedingt vor der Bildung der Lösungsvarianten eine Untersuchung der Softwarekomponenten hinsichtlich des Beitrags zu den Funktionsgruppen durchgeführt werden.

Bei der Zuordnung der Softwarekomponenten zu den Funktionen wurde grob geprüft, ob und wie diese die Funktion erfüllen. Dazu wurden Demonstrationen durch die Vendoren der Softwarekomponenten und vorhandene Testsysteme genutzt. So konnten Schwachstellen von resultierenden Architekturvarianten identifiziert und beseitigt oder zumindest gegenüber anderen Lösungen abgewogen werden. Letztlich ergaben sich so zwei verschiedene Lösungsvarianten. Variante 1 basierte auf dem vorhandenen PDM-System, welches die grundlegenden Funktionen zur Speicherung der Daten übernehmen sollte. Da bereits ein

weiteres, allgemein akzeptiertes Werkzeug zur Anforderungsverwaltung im Unternehmen existierte, wurde dieses beibehalten, um Anforderungen zu erstellen und zu bearbeiten. Beide Softwarekomponenten können jedoch nur unzureichend den Import von Kundenanforderungen umsetzen, sodass hierzu eine separate Softwarekomponente ausgewählt wurde. Um Architekturmodelle zu erstellen, wurde ein SysML-Modellierungswerkzeug ausgewählt, welches den gleichen Vendor wie das PDM-System aufweist. Eine eng in das Modellierungswerkzeug integriertes TDM-System übernimmt die Verwaltung der Architekturobjekte. Variante 2 basierte vollständig auf der Lösung des PDM-Anbieters und kam so ohne Schnittstellen zu externen Sytemen aus, wurde jedoch kritisch hinsichtlich der Anforderungserfüllung und der Gebrauchstauglichkeit gesehen. Bild 6-9 zeigt im Kontext der dargestellten Architektur, welche Softwarekomponenten welche Funktionen umsetzen.

Allokations-Matrix V1
Fragestellung: „Welche Funktion (Zeilen) wird durch welche Softwarekomponente (Spalten) umgesetzt?“
Legende Anf.: Anforderung, Arch.: Architektur

	SW-K1	SW-K2.1	SW-K2.2	SW-K2.3	SW-K3
F Anf.-Objekte administrieren			X		
F Anf.-Objekte bearbeiten			X		
F Anf.-Objekte speichern		X			
F Anf.-Lebenszyklus steuern		X			
F Anf.-Reports erstellen			X		
F Kundenanf. importieren				X	
F Arch.-Objekte bearbeiten					X
F Arch.-Objekte speichern	X				
F Arch.-Lebenszyklus steuern		X			

Allokations-Matrix V2
Fragestellung: „Welche Funktion (Zeilen) wird durch welche Softwarekomponente (Spalten) umgesetzt?“
Legende Anf.: Anforderung, Arch.: Architektur

	SW-K1	SW-K2.1	SW-K2.2	SW-K2.3	SW-K3
F Anf.-Objekte administrieren			X		
F Anf.-Objekte bearbeiten			X		
F Anf.-Objekte speichern			X		
F Anf.-Lebenszyklus steuern			X		
F Anf.-Reports erstellen			X		
F Kundenanf. importieren			X		
F Arch.-Objekte bearbeiten					X
F Arch.-Objekte speichern					X
F Arch.-Lebenszyklus steuern					X

Bild 6-9: Allokationsmatrix zur Darstellung der Funktionen und ihrer Umsetzung durch Softwarekomponenten für die verschiedenen Lösungsvarianten V1 und V2

Komponentenarchitektur

Im weiteren Prozess wurden im Sinne der Wirtschaftlichkeit nur die zwei zuvor ausgewählten Varianten weiter betrachtet. Im Rahmen der Validierung wurde die Komponentenarchitektur parallel zur Definition der Lösungsvarianten entwickelt. Dies war sinnvoll, da so unmittelbar geprüft werden konnte, ob die Konsistenz einer Lösungsvariante auch über die Schnittstellen abbildbar ist. Dazu wurden bei der Erstellung einer Lösungsvariante auch eine erste Architektur entwickelt, wodurch sichergestellt wurde, dass überhaupt entsprechende Schnittstellen zwischen Softwarekomponenten verfügbar sind. Hierzu wurden entsprechende Ports modelliert und der Schnittstellentyp und die Offenheit festgelegt. Ergibt sich eine konsistente und vielversprechende Architekturvariante, werden die Schnittstellen und Flüsse, wie in Bild 6-10 ausschnittweise dargestellt, vollständig spezifiziert und auch hinsichtlich der folgenden Evaluierung gewichtet.

Das Ergebnis zeigt, dass im Rahmen der Lösungsfindung auch komplexere Lösungsvarianten darstellbar sind. Die dargestellte Lösungsvariante nutzt ein spezielles Parsing Werkzeug (SW-K2.3), um PDF-basierte Kundenanforderungen einzulesen und per ReqIF-Anbindung

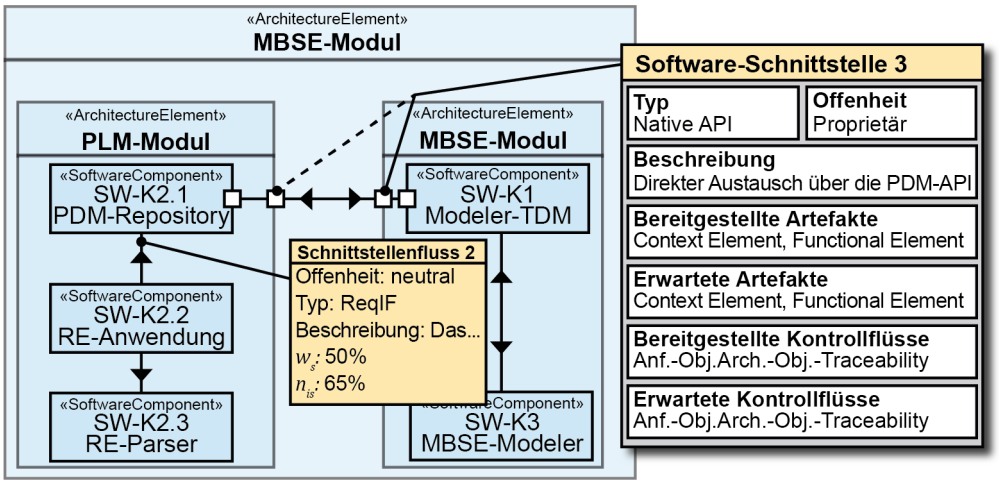


Bild 6-10: Ausschnitt aus der Komponentenarchitektur mit der beispielhaften Spezifikation von Schnittstellen

an eine bereits vorhandene RE-Anwendung (SW-K2.2) zu senden. Diese ist das primäre Autorenwerkzeug für die Anwender. Um die Rückverfolgbarkeit sicherzustellen, werden alle Anforderungen mit einer ReqIF-Schnittstelle mit dem PDM-System (SW-K2.1) ausgetauscht und durch dieses gespeichert und verwaltet. Dies gilt auch für alle Architekturobjekte, die das MBSE-Modellierungswerkzeug (SW-K3) über die native Schnittstelle mit dem dazugehörigen TDM-System austauscht. Alle Anforderungs- und Architekturobjekte können in dieser Lösungsvariante zentral durch das PDM-System kontrolliert werden, z. B. zur Verwaltung der Rückverfolgbarkeit oder der Konfigurationskontrolle. Eine zwingende Nutzung einer neuen und ggf. funktional oder hinsichtlich der Gebrauchstauglichkeit weniger geeigneten RE-Anwendung des PDM-Anbieters ist somit nicht notwendig.

Während der Erarbeitung der Modelle des Anwendungsbeispiels wurden kontinuierlich die Verknüpfungen zur Sicherstellung der Rückverfolgbarkeit gepflegt. Dies erfolgte unmittelbar bei der Erstellung der Modelle, da andernfalls die Gefahr bestand, nicht alle Verknüpfungen zu erfassen. Zudem wurden nach jedem Modellierungsschritt Reviews mit dem beteiligten Team und weiteren Stakeholdern durchgeführt, um die Vollständigkeit und Korrektheit der Modellinhalte sicherzustellen.

Evaluierung der Lösungsvarianten

Für jede der beiden Lösungsvarianten wurden die Anforderungen hinsichtlich der funktionalen Eignung bewertet. Dies geschah bei bereits bekannten Softwarekomponenten durch Tests in einem vorhandenen System. Bei noch nicht vorhandenen Softwarekomponenten wurden die Funktionen in Workshops durch die Softwarevendoren detailliert demonstriert und dabei ihr Erfüllungsgrad durch Prozess-, Methoden- und IT-Verantwortliche bewertet. Die Dokumentation erfolgt zunächst in Papierform auf Basis eines Exports des Lösungsmodells. Die aggregierte Bewertung über alle Bewertenden für die Lösungsvariante 1 wurde dann in das rechnerbasierte Lösungsmodell übertragen und ist in Tabelle 6-1 dargestellt.

Tabelle 6-1: Ausschnitt aus der Bewertung der funktionalen Eignung

Kriterium	IT-Werk-zeug	Kriteriengewicht		Teilnutzwert	
		w'_k	w_k (norm.)	n'_{ik}	n_{ik} (norm.)
FG Systemkonzipierung unterstützen		100%	100%	87%	87,000%
FG Anforderungen handhaben		50%	50%	83%	41,500%
F Anf.-Objekte administrieren	SW-K1	15%	7,5%	90%	6,750%
BK Rollen & Rechte definieren	SW-K1	30%	2,25%	90%	2,250%
BK Rollenzuweisung in Tabellenansicht	SW-K1	10%	0,75%	30%	0,225%
BK Anf.-Objekt Typen definieren	SW-K1	30%	2,25%	100%	2,250%
BK Anf.-Objekt Attribute definieren	SW-K1	30%	2,25%	100%	2,250%
F ...	SW-K4	20%	10,5%	77%	8,085%

Legende FG Funktionsgruppe F Funktion BK Bewertungskriterium norm.: normiert

Bezüglich der Bewertung der Gebrauchstauglichkeit wurden UEQ-Tests für die Anwendungsfälle durch potentielle Nutzer durchgeführt. Das aggregierte Ergebnis wurden ebenfalls in das Lösungsmodell eingepflegt (vgl. Tabelle 6-2).

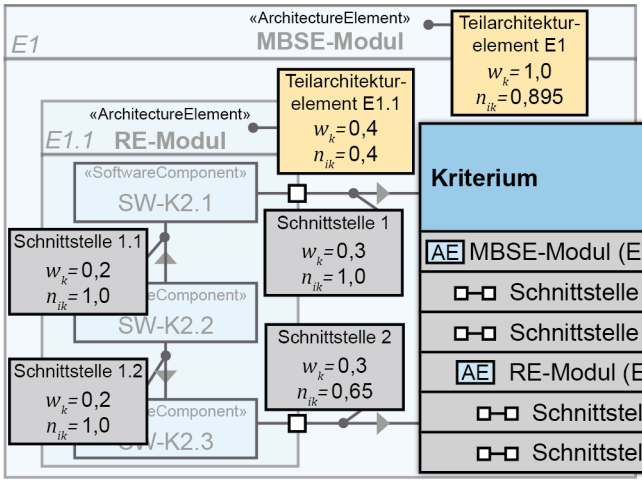
Tabelle 6-2: Ausschnitt aus der Bewertung der User Experience

Kriterium	IT-Werk-zeug	Kriteriengewicht		UEQ Wert	Teilnutzwert	
		w'_k	w_k (norm.)		n'_{ik}	n_{ik} (norm.)
FG Systemkonzipierung unterstützen		100%	100%		87%	73,413%
FG Anforderungen handhaben		50%	50%		83%	37,613%
F Anf.-Objekte administrieren	SW-K1	15%	7,5%		90%	5,693%
UC Neue Rolle anlegen	SW-K1	50%	2,25%	2,73	95,5%	3,339%
UC Neuen Anf.-Objekt Typen anlegen	SW-K1	50%	0,75%	-0,23	46,2%	2,354%
F ...	SW-K4	20%	10,5%		77%	7,125%

Legende FG Funktionsgruppe F Funktion UC Anwendungsfall norm.: normiert

Die Bewertung der Interoperabilität ergab sich aus der Komponentenarchitektur, die parallel zum Lösungsmodell erstellt wurde, um die Konsistenz der Lösungsvarianten sicherzustellen. Dazu wurde die Struktur bottom-up in einem interdisziplinären Team aus Methoden- bzw. Prozessspezialisten und IT-Spezialisten bewertet. Ein Ausschnitt des Resultates ist in Bild 6-11 dargestellt.

Als ergänzende Anforderung wurde auf Wunsch des Unternehmens die Wartbarkeit durch die Engineering-IT berücksichtigt. Der Nutzwert wurde durch den IT-Bereich bestimmt und in die Nutzwertanalyse übernommen. Durch die aggregierten Nutzenwerte der verschiedenen Bewertungsdimensionen ergab sich letzlich eine objektive und transparente Entscheidungsgrundlage für das Management. Auch für andere Stakeholder war die Entscheidung so nachvollziehbar, auch wenn diese ggf. nicht vollständig die eigenen Interessen widergespiegelt hat.



Kriterium	Kriterien-gewicht	Teilnutzwert	
	w_k(norm.)	n'_{ik}	n_{ik}(norm.)
AE MBSE-Modul (E1)	1,0	0,895	0,895
Schnittstelle 1	0,3	1,000	0,300
Schnittstelle 2	0,3	0,650	0,195
AE RE-Modul (E1.1)	0,4	1,000	0,400
Schnittstelle 1.1	0,2	1,000	0,200
Schnittstelle 1.2	0,2	1,000	0,200

Bild 6-11: Bewertung der Interoperabilität einer Architekturvariante

6.2 Bewertung der Arbeit an den Anforderungen

Nachfolgend wird die entwickelte *Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen* anhand der in Kap. 3.6 definierten Anforderungen bewertet und die Validierung der Arbeit kritisch betrachtet. Bild 6-12 stellt den Bezug der Anforderungen auf die Bestandteile der Spezifikationstechnik dar.

A1) Fokus auf das Engineering: Die entwickelte Spezifikationstechnik ist auf die Anwendung im Engineering ausgelegt. Dazu berücksichtigt das Vorgehensmodell die Modellierung von Prozessen, Methoden, Modellen sowie IT-Werkzeugen im Kontext des Engineerings (vgl. Kap. 5.2). Es leitet für diesen spezifischen Anwendungsfall den Nutzer zur Anwendung an. In Verbindung mit der Modellierungssprache erlaubt das Vorgehensmodell eine tiefgreifende methodische Unterstützung mit domänespezifischen Sprachkonstrukten (vgl. Kap. 5.3.2). Die Anpassung domäneunspezifischer Modellierungssprachen oder -rahmenwerke entfällt somit. Auch die zur Modellierung sinnvollen Sichtweisen sind bereits definiert und erlauben so eine effiziente und fokussierte Anwendung der Modellierungssprache (vgl. Kap. 5.3.1.3).

A2) Analyse der Prozess-, Methoden- und Datenarchitektur: Die Spezifikationstechnik adressiert die Analyse der Prozess-, Methoden- und Datenarchitektur in zwei Partialmodellen. Das Prozessmodell beschreibt Aktivitäten, Artefakte und ergänzende Informationen auf Prozessebene (vgl. Kap. 5.3.2.2). Das Methodenmodell fokussiert Tätigkeiten der Methoden und hierarchisiert die Artefakte aus der Prozessebene bis zur Attributebene auf Basis der Methode. Zudem werden die Artefaktbeziehungen analysiert (vgl. Kap. 5.3.2.3). Eine fokussierte Architektur der Daten kann zudem über die Viewpoints *PMV03 – Artefaktdekomposition* sowie *PMV 04b – Artefaktbeziehungen* abgebildet werden. Weitere Viewpoints unterstützen die Analyse einzelner Aspekte (vgl. Kap. 5.3.1.3 bzw. A2). Der Anwender wird im Vorgehensmodell durch die Phasen 1 und 2 zur Durchführung einer Analyse angeleitet (vgl. Kap. 5.2.1 und 5.2.2).

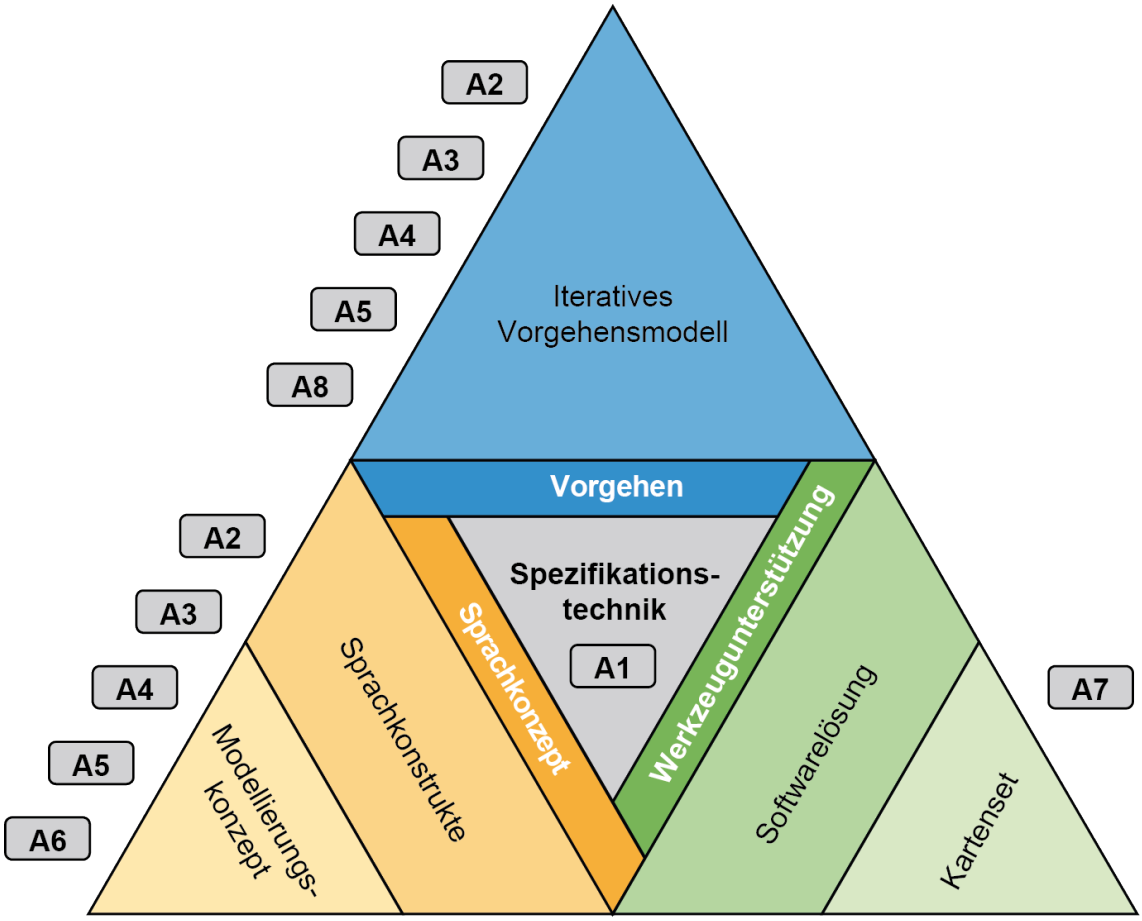


Bild 6-12: Erfüllung der Anforderungen durch die Spezifikationstechnik

A3) Identifikation und Analyse von Anforderungen: Die Identifikation und Analyse wird durch die Analyse-Partialmodelle unterstützt. Sie erlauben eine schrittweise Detaillierung von Prozessen über Methoden hin zu Anwendungsfällen (vgl. Kap. 5.3.2.2 bis 5.3.2.4) und Anforderungen. Die im Rahmen des Engineerings durchzuführenden Tätigkeiten, resultierende Artefakte und Relationen können systematisch untersucht und relevante Anforderungen abgeleitet werden. Der Betrachtungsfokus kann über das Sichtenkonzept gezielt gesteuert werden (vgl. Kap. 5.3.1.3). Die Dokumentation erfolgt über das Partialmodell Anforderungen (vgl. Kap. 5.3.2.6). Die Strukturierung der Anforderungen wird über die Funktionshierarchie unterstützt.

A4) Ableiten geeigneter Applikationsarchitekturen: Die Aufteilung der Architektur in eine funktionale und eine Komponentenarchitektur unterstützt die schrittweise Herleitung zur Umsetzung des Zielbildes. Die funktionale Architektur zeigt notwendige Schnittstellen auf, ohne die technische Lösung unnötig einzuschränken. Um die systematischen Ableitung der Elemente und Beziehungen der funktionalen Hierarchie zu unterstützen, werden Heuristiken bereitgestellt (vgl. Kap. 5.3.2.5). Die angeleitete Bildung technischer Lösungen wird gezielt durch die Morphologie des Lösungsmodells unterstützt (vgl. Kap. 5.3.2.7).

A5) Anleitung zur Architektur-Evaluierung: Das Vorgehensmodell stellt sicher, dass ein geeignetes Zielsystem zur Bewertung einer Anwendungsarchitektur für das Engineering abgeleitet wird (vgl. Kap. 5.2.3). Dabei werden durch das Sprachkonzept geeignete Kriterien und Modellelemente zur Spezifizierung der drei Dimensionen der funtionalen Eignung, der Gebrauchstauglichkeit und der Interoperabilität bereitgestellt. Zudem können bei Bedarf eigene Bewertungsdimensionen berücksichtigt werden (vgl. Kap. 5.3.2.7).

A6) Modellierungsrahmenwerk zur Spezifikation: Die Spezifikation der Engineering-IT-Architektur erfolgt über eine Modellierungssprache mit definierter Syntax und Semantik. Das Metamodell legt über die notwendigen Sprachelemente und ihre Zusammenhänge die abstrakte Syntax und Teile der statischen Semantik fest (vgl. Kap. 5.3.1.1). Die vollständige Semantik und die konkrete Syntax sind textuell und in Tabellen mit den Sprachkonstrukten definiert (vgl. Kap. 5.3.2). Die Sprachkonstrukte sind explizit zur Beschreibung von Prozessen, Methoden, Modellen und IT-Werkzeugen im Engineering entwickelt worden. Sie sind zudem durch das Farbkonzept und die Nutzung differenzierter Formen für die verschiedenen an der Modellierung beteiligten Rollen verständlich und einprägsam. Das Modellierungskonzept definiert im Sinne eines Modellierungsrahmenwerks grundlegende Konzepte zur Erstellung, Analyse und Verwendung der Modelle in der Praxis. Dies betrifft insbesondere die systematische Nutzung von definierten Sichten (vgl. Kap. 5.3.1.2, 5.3.1.3) und die Vernetzung der Partialmodelle (vgl. Kap. 5.3.1.4). Dadurch entsteht ein integriertes Modell, welches alle relevanten Aspekte umfasst.

A7) Werkzeugunterstützung: Um das Sprachkonzept in der Praxis für die Erstellung von Modellen nutzen zu können, wird eine Werkzeugunterstützung bereitgestellt. Ein papierbasiertes Kartenset erlaubt insbesondere die Darstellung der Notation im Rahmen erster Workshops zur Erfassung von Modellinhalten (vgl. Kap. 5.4.1). Die rechnergestützte Modellierung wird durch eine Softwarelösung ermöglicht. Dazu wurde ein Sprachprofil erstellt, welches bestehende Modellierungssprachen um die spezifischen Sprachkonstrukte der Spezifikationstechnik erweitert und so die Modellierung in verschiedenen Modellierungswerkzeugen erlaubt. Zudem formalisiert es das Sprachkonzept durch die Definition von Attributen (vgl. Kap. 5.4.2). Es wurde prototypisch als Plug-in in ein kommerzielles Modellierungswerkzeug implementiert. Dabei wurden auch die Notation und die für die Darstellung der Sichten notwendigen Diagramme berücksichtigt (vgl. Kap. 5.4.3). Dies erlaubt den Modellierenden, die Spezifikationstechnik ohne tiefe Kenntnisse der zugrundeliegenden Modellierungssprachen komfortabel und zielgerichtet anzuwenden.

A8) Vorgehensmodell: Das Vorgehensmodell unterstützt die Anforderung unter Anwendung der Sprachkonstrukte. Das Vorgehen vermittelt dem Anwender der Spezifikationstechnik einen Überblick über die zu durchlaufenden, sachlogisch geordneten Phasen zur Erstellung einer Engineering-IT-Architektur. Dabei sind die durchzuführenden Tätigkeiten und die zu erarbeitenden Resultate aufgeführt. Diesen sind Hilfsmittel und Partialmodelle zugeordnet (vgl. Kap. 5.2). Die Nutzung der Hilfsmittel und Hinweise zur Anwendung der Modellierungssprache sind im Rahmen der Beschreibung der Partialmodelle beschrieben (vgl. Kap. 5.3.2).

7 Zusammenfassung, Reflexion und Ausblick

Technische Erzeugnisse erleben einen Wandel von rein mechanischen oder mechatronischen Systemen hin zu intelligenten technischen Systemen. Diese führen zu verschiedenen Herausforderungen, u. a. wachsender Interdisziplinarität, einer hohen Schnittstellenvielfalt, steigender Anforderungskomplexität und zunehmender Produktvarianz. Diese enorme Komplexität der ITS führt dazu, dass auch ihre Entwicklung, das Engineering, komplexer wird und sich an die Herausforderungen anpassen muss. Die Antwort darauf sind Ansätze wie das Smart Engineering oder Advanced Systems Engineering.

Diese Ansätze führen zu **Veränderungen** in den Prozessen, Methoden oder Modellen im Engineering. In der Praxis zieht dies eine steigende Anzahl von Prozessen und Methoden nach sich. Entwicklungsartefakte werden zudem aufgrund der Vielzahl beteiligter Domänen sehr heterogen. Die Digitalisierung verstärkt diesen Effekt, da immer mehr Artefakte durch formale, strukturierte Informationsmodelle repräsentiert werden sollen. Ziel ist oftmals die Erhöhung der Durchgängigkeit im Engineering durch eine optimierte Datenverfügbarkeit oder vernetzte Artefakte, z. B. aufgrund geforderter Rückverfolgbarkeit. Dies führt dazu, dass die im Engineering genutzten einzelnen IT-Werkzeuge diesen Anforderungen nicht mehr gerecht werden. Vielmehr sind dafür oftmals über Schnittstellen vernetzte Systeme von IT-Werkzeugen notwendig, die jedoch hinsichtlich Funktionalität, Gebrauchstauglichkeit und Interoperabilität unterschiedliche Eigenschaften aufweisen können.

Für die Engineering-IT ergibt sich daher die **Problemstellung**, welche Anforderungen aus den veränderten Prozessen, Methoden und Modellen an eine geeignete IT-Lösung resultieren. Darüber hinaus besteht die Fragestellung, wie eine geeignete Applikationsarchitektur gestaltet und hinsichtlich der Erfüllung der Anforderungen bewertet werden kann. Dabei bietet sich die Nutzung eines integrierten Modells zur Analyse und Spezifikation der Engineering-IT-Architektur an. Hierzu bedarf es eines systematischen und methodischen Vorgehens, welches auf die Bedingungen im Engineering zugeschnitten ist und den Anwender anleitet.

Im Rahmen der vorliegenden Arbeit wurden daher **existierende Ansätze** zur ganzheitlichen Beschreibung von Architekturen untersucht. Hier sind insbesondere die Modellierungssprachen UML/SysML, BPMN und SPEM zu erwähnen, welche kombiniert die Grundlage zur Beschreibung von Prozessen, Methoden und Architekturen erlauben. Darüber hinaus sind die genannten Sprachen über einen Profilmechanismus mittels spezifischer Sprachkonstrukte erweiterbar. Weiterhin wurden Ansätze zur Analyse von Anforderungen und Prozessen untersucht. Es wurden zudem Ansätze zur Definition von Enterprise und IT-Architekturen sowie zur Auswahl von IT-Werkzeugen betrachtet. Das TOGAF-Framework stellt einen vielversprechenden Ansatz mit einem Vorgehensmodell zur Modellierung von Enterprise Architekturen dar. Die FAS-Methode stellt einen methodischen Ansatz zur Ableitung funktionaler Architekturen dar. Zur Auswahl von IT-Werkzeugen bieten sich

auf der Nutzwertanalyse basierende Ansätze wie das 3-Phasen-Konzept nach SCHUH an. Die analysierten Ansätze unterstützen nur einzelne Aspekte bei der modellbasierten Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen. Eine ganzheitliche Spezifikationstechnik, welche alle gestellten Anforderungen erfüllt, konnte nicht ermittelt werden. Somit bestand **Handlungsbedarf** für eine Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen.

Die erarbeitete Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen greift diesen Handlungsbedarf auf. Diese nimmt Grundlagen der untersuchten Ansätze auf, überträgt sie auf den Anwendungsfall der Engineering-IT-Architektur und erweitert diese um neu entwickelte Konstrukte und Hilfsmittel. Das **Resultat** ist die Spezifikationstechnik, welche sich in die drei Bestandteile Vorgehensmodell, Sprachkonzept sowie Werkzeugunterstützung gliedert.

- Das Vorgehensmodell leitet den Anwender zur Erstellung einer Engineering-IT-Architektur mit der Spezifikationstechnik an. Es definiert die Phasen und durchzuführende Tätigkeiten und benötigte Hilfsmittel.
- Das Sprachkonzept definiert im Sinne eines Modellierungsrahmenwerks ein *Modellierungskonzept*, welches grundlegende Konzepte zur Erstellung, Analyse und Verwendung der Modelle umfasst. Dazu gehört auch ein Ansatz, um systematisch mit definierten Sichten auf eine Engineering-IT-Architektur zu arbeiten. Ebenso wurde ein Ansatz für die sichtenübergreifenden Beziehungen innerhalb des Modells festgelegt. Zur Erstellung der Modelle wird eine an die Spezifikation von Engineering-IT-Architekturen angepasste Modellierungssprache mit entsprechenden *Sprachkonstrukten* bereitgestellt. Die abstrakte Syntax und Teile der statischen Semantik sind über ein Metamodell definiert. Die konkrete Syntax und die Semantik sind im Rahmen von Kap. 5.3.2 textuell beschrieben. Zudem wird die Nutzung der Sprachkonstrukte und Hilfsmittel beschrieben.
- Die Werkzeugunterstützung unterstützt die praktische Anwendung des Sprachkonzepts zur Erstellung und Nutzung von Modellen. Ein *Kartenset* erlaubt die Nutzung der Sprachkonstrukte mit ihrer konkreten Syntax im Rahmen von Workshops. Die *Softwarelösung* ermöglicht die rechnerbasierte Repräsentation des Modells und ist für die effiziente Nutzung und Wartung des Modells notwendig. Sie beinhaltet einerseits ein Sprachprofil zur Umsetzung der Modellierungssprache auf Basis der UML, SysML, BPMN und SPEM mit domänespezifischen Stereotypen. Das Metamodell definiert die Modellierungssprache und ist rechnerinterpretierbar. Die Softwarelösung beinhaltet eine prototypische Anpassung eines kommerziellen Modellierungswerkzeugs, bei dem das Sprachprofil und die entsprechenden Diagrammtypen implementiert wurden und die Benutzeroberfläche auf die Anwendung der Spezifikationstechnik angepasst wurde.

Die **Validierung** der Spezifikationstechnik erfolgte im Rahmen verschiedener Projekte, mit dem Ziel der Integration der Methode in den Entwicklungsprozess, der Harmonisierung

von Methoden und Artefakten in den unterschiedlichen Unternehmensbereichen und der Auswahl einer geeigneten IT-Applikationsarchitektur. Inhaltlich wurden zwei Domänen des Systems Engineering eines Landtechnikkonzerns betrachtet, nämlich das Management textueller Anforderungen sowie grafischer Architekturmodelle. Dabei wurden alle Schritte des Vorgehensmodells durchlaufen und das Sprachkonzept sowie die Werkzeugunterstützung angewandt. Dadurch konnte aufgezeigt werden, dass alle Anforderungen der Arbeit vollständig erfüllt werden.

Zur Sicherung der wissenschaftlichen Qualität und Objektivität wurde die Arbeit nach dem im Kap. 2.3 beschriebenen Vorgehen durchlaufen, welches sich an der DRM nach BLESSING UND CHAKRABARTI und der Konsortialforschung nach ÖSTERLE UND OTTO orientiert. Alle vier Phasen des Vorgehens wurden vollständig durchlaufen. In der ersten Phase wurde die Forschungsfrage umrissen (s. Kap. 1). In der zweiten Phase wurde die Forschungsfrage untermauert und konkretisiert sowie qualitative Anforderungen an die Arbeit abgeleitet (s. Kap. 3). Zudem wurden die wissenschaftlichen Grundlagen erarbeitet um den Wissensstand zu analysieren (s. Kap. 4). In der dritten Phase wurde die Erstellung der Design Artefakte und der dazu notwendigen Schritte beschrieben (s. Kap. 5). Die dritte Phase beschreibt schließlich die Evaluation am oben genannten Anwendungsbeispiel. Der Erfolg des erarbeiteten Lösungsansatzes wird anhand der Erfüllung der Anforderungen bewertet (s. Kap. 6).

Die Evaluierung wurde dabei iterativ über alle Phasen der DRM durchgeführt. Die Anwendung im Unternehmen an einem realen und repräsentativen Beispiel hat während der Entwicklung der Spezifikationstechnik Schwächen aufgezeigt, die in der vorliegenden Arbeit beseitigt werden konnten. Dadurch wurde auch die Praxistauglichkeit der Arbeit sichergestellt, sowohl hinsichtlich der Relevanz der Problemstellung als auch des Beitrages des entwickelten Lösungsansatzes. Aufgrund der zur Verfügung stehenden personellen Ressourcen im Unternehmen wurde die Evaluierung qualitativ durchgeführt. Die Evaluierung der Gebrauchstauglichkeit erfolgte durch iteratives Feedback der Anwender.

Basierend auf den Erfahrungen der Validierung sollen **Limitationen** der Arbeit erwähnt werden. Die Anwendung der Arbeit für einen größeren Aufgabenkomplex ist relativ aufwändig, da das notwendige Wissen oftmals nicht explizit vorliegt und das Wissen somit zunächst extrahiert werden muss. Auch die Pflege des Modells bedarf aufgrund des hohen Vernetzungsgrades eines nicht zu unterschätzenden Aufwandes. In der praktischen Anwendung ist jedoch auch klargeworden, dass ohne die Nutzung der Spezifikationstechnik dieses implizite Wissen gar nicht verfügbar ist, das Engineering nicht ganzheitlich betrachtet werden kann und die Auswirkungen von Änderungen aufgrund der fehlenden Traceability kaum abgeschätzt werden können. Somit bestätigt sich hier schließlich die Aussage von TÖRGREN UND GRONAN [TG18, S. 7], dass sich die zunehmende Komplexität der betrachteten Systeme (des Engineerings) auf die Komplexität ihrer Entwicklung auswirkt (der Spezifikationstechnik). Das Nutzenverhältnis konnte jedoch positiv bewertet werden. Dies setzt allerdings voraus, dass das Gesamtmodell von einem Experten verwaltet und die Anwendungs des Vorgehensmodells und Sprachkonzeptes koordiniert wird. Dies hat sich

in der praktischen Anwendung als notwendig erwiesen, damit die Modellierungsregeln eingehalten werden und somit die Konsistenz sichergestellt ist. Die Arbeit wurde zwar in verschiedenen Domänen und Unternehmensbestandteilen durchgeführt und deckt dadurch mit hoher Wahrscheinlichkeit die Anforderungen für eine generische Nutzung, mindestens in der Landtechnik, ab.

Aus der Arbeit ergibt sich somit zukünftiger **Forschungsbedarf**. Durch die Anwendung in weiteren Branchen kann die Arbeit hinsichtlich einer generischen Nutzbarkeit weiter validiert werden. Hinsichtlich der Spezifikationstechnik ist zudem zu prüfen, ob die Anwendbarkeit ggf. durch weitere Hilfsmittel verbessert werden kann. Denkbar wäre hier bspw. ein Funktionskatalog mit typischen und häufig wiederkehrenden Funktionen zur Nutzung in funktionalen Architekturen. Darüber hinaus sollte geprüft werden, ob ggf. weitere Heuristiken zur methodischen Anleitung der Nutzer bei der Erstellung von Engineering-IT-Architekturen identifiziert werden können. Dies würde den Ansatz weiter systematisieren. Ferner ist eine Erweiterung der Bewertungskriterien zu prüfen. Dies betrifft die Integration der nicht betrachteten, technischen Bewertungsdimensionen der ISO/IEC 25010 in die Evaluierung. Die ISO/IEC 25010 betrachtet zudem nicht den Aspekt der Kosten. Dieser Aspekt ist im Hinblick auf sehr unterschiedliche Applikationsarchitekturen schwierig zu bewerten, da unterschiedliche IT-Werkzeuge unterschiedliche Kostenprofile für Lizenz-, Wartungs-, Anpassungs- oder Servicekosten aufweisen. Diese können außerdem von zahlreichen weiteren Parametern abhängig sein. Es ist daher sinnvoll, zu untersuchen, wie die Spezifikationstechnik einen Beitrag zur Ermittlung der perspektivischen Gesamtkosten über die Nutzungsdauer leisten kann und wie diese Berechnung in die Spezifikationstechnik integriert werden kann. Die Infrastrukturebene wurde im Rahmen der Arbeit nicht berücksichtigt, was im Rahmen der Validierung jedoch keine negativen Auswirkungen hatte und im Bedarfsfall durch die Nutzung allgemeiner EAM-Ansätze ausgleichbar wäre.

Abschließend ist anzumerken, dass die Werkzeuge und ihre Interoperabilität eine grundsätzliche Limitierung darstellen: Wenn keine Werkzeuge mit kompatiblen Schnittstellen zur Lösung einer funktionalen Architektur bereitstehen, besteht prinzipiell die Gefahr, dass die Spezifikationstechnik nicht zu praktisch umsetzbaren Komponentenarchitekturen führt. Obgleich in der Vergangenheit mit Initiativen wie ReqIF oder OSLC bereits Maßnahmen zur Standardisierung der Schnittstellen zwischen IT-Werkzeugen unternommen wurden, ist hier weitere Forschung notwendig. Insbesondere sollten hier auch standardisierte Funktionsmerkmale für die Schnittstellen festgelegt werden und in die Spezifikationstechnik einfließen. Dadurch kann die Eignung einer Schnittstelle für einen spezifischen Fall einfacher bewertet werden.

Abkürzungsverzeichnis

ADM	Architecture Development Method
AHP	Analytic Hierarchy Process
ALM	Application Lifecycle Management
API	Application Programming Interface
ASE	Advanced Systems Engineering
BPMN	Business Process Model and Notation
bspw.	beispielsweise
bzw.	beziehungsweise
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CASE	Computer-Aided Software Engineering
CM	Konfigurationsmanagement/Configuration Management
CONSENS	Conceptual design Specification technique for the Engineering of complex Systems
COTS	Commercial of the Shelf
CPS	Cyber-Physical System
CRUD	Create, Read, Update, Delete
DRM	Design Research Methodology
EA	Enterprise Architektur/Enterprise Architecture
EAM	Enerprise Architecture Management
ERP	Enerprise Resource Planning
FAS	Funktionale Architekturen für Systeme
ggf.	gegebenenfalls
IS	Informationssystem
IT	Informationstechnologie
ITS	Intelligentes technisches System

Kap.	Kapitel
MBSE	Model-based Systems Engineering
NWA	Nutzwertanalyse
PDM	Product Data Management
PLM	Product Lifecycle Management
PLZ	Produktlebenszyklus
PPS	Produktionsplanung und -steuerung
OCL	Object Constraint Language
o. g.	oben genannt
OMG	Object Management Group
OMEGA	Objektorientierte Methode zur Geschäftsprozessmodellierung und -analyse
OSLC	Open Services for Lifecycle Collaboration
RE	Requirements Engineering
ReqIF	Requirements Interchange Format
s.	siehe
SE	Systems Engineering
SPEM	Software & Systems Process Engineering Metamodel
STEITA	Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering IT-Architekturen
SysML	Systems Modeling Language
TDM	Team Data Management
TOGAF	The Open Group Architecture Framework
u. a.	unter anderem
UEQ	User Experience Questionnaire
UML	Unified Modeling Language
UX	User Experience
vgl.	vergleiche
z. B.	zum Beispiel

Literaturverzeichnis

Publikationen

[AAP+19] ARMENDIA, M.; ALZAGA, A.; PEYSSON, F.; FUERTJES, T.; CUGNON, F.; OZTURK, E.; FLUM, D.: Machine Tool – From the Digital Twin to the Cyber-Physical Systems. In: Armendia, M.; Ghassempouri, M.; Ozturk, E.: Twin-Control. Springer, 2019, pp. 3–21 – ISBN: 978-3-030-02202-0, DOI: 10.1007/978-3-030-02203-7_1

[Abr18] ABRAMOVICI, M.: Engineering smarter Produkte und Services – Plattform Industrie 4.0 Studie, März 2018

[Ada96] ADAM, D.: Planung und Entscheidung – Modelle, Ziele, Methoden. Gabler, 4. Auflage, 1996 – ISBN: 3-409-44613-3

[ADE+11] ARNOLD, V.; DETTMERING, H.; ENGEL, T.; KARCHER, A.: Product Lifecycle Management beherrschen. Springer, 2. Auflage, 2011 – ISBN: 978-3-642-21812-5, DOI: 10.1007/978-3-642-21813-2

[AES+12] ANDERL, R.; EIGNER, M.; SENDLER, U.; STARK, R. (Hrsg.): Smart Engineering – Interdisziplinäre Produktentstehung. Springer Vieweg, 2012 – ISBN: 978-3-642-29371-9, DOI: 10.1007/978-3-642-29372-6

[AG12a] ABRAMOVICI, M.; GÖBEL, J. C.: Decision Support Framework for PLM Harmonization Projects within Industrial Companies. In: Rivest, L.; Bouras, A.; Louhichi, B.: Product Lifecycle Management – Towards Knowledge-Rich Enterprises – IFIP WG 5.1 International Conference, PLM 2012, Montreal, QC, Canada, July 9-11, 2012. Springer, 2012, pp. 535–546 – ISBN: 978-3-642-35757-2, DOI: 10.1007/978-3-642-35758-9_48

[AG12b] ALBERS, A.; GAUSEMEIER, J.: Von der fachdisziplinorientierten Produktentwicklung zur Vorausschauenden und Systemorientierten Produktentstehung. In: Anderl, R.; Eigner, M.; Sendler, U.; Stark, R.: Smart Engineering - Interdisziplinäre Produktentstehung. Springer Vieweg, Berlin, 2012, S. 17–29 – ISBN: 978-3-642-29371-9, DOI: 10.1007/978-3-642-29372-6_3

[AH16] ABRAMOVICI, M.; HERZOG, O. (Hrsg.): Engineering im Umfeld von Industrie 4.0 – Einschätzungen und Handlungsbedarf. Herbert Utz Verlag, München, 2016 – ISBN: 978-3-8316-4501-5

[All15] ALLWEYER, T.: BPMN 2.0 - Business Process Model and Notation – Einführung in den Standard für die Geschäftsprozessmodellierung. Books on Demand, 3. Auflage, 2015 – ISBN: 978-3-7386-2671-1

[Alt12] ALT, O.: Modellbasierte Systementwicklung mit SysML. Carl Hanser, 2012 – ISBN: 978-3-446-43066-2

[And18] ANDERL, R.: Virtuelle Produktentstehung. In: Grote, K.-H.; Bender, B.; Göhlich, D.: Dubbel. Springer Vieweg, 25. Auflage, 2018, Y18–Y34 – ISBN: 978-3-662-54804-2, DOI: 10.1007/978-3-662-54805-9_157

[AZL19] ALVAREZ-RODRÍGUEZ, J. M.; ZUÑIGA, R. M.; LLORENS, J.: Elevating the meaning of data and operations within the development lifecycle through an interoperable toolchain. INCOSE International Symposium 29 (1), 2019, pp. 1053–1071

[BC09] BLESSING, L. T.; CHAKRABARTI, A.: DRM, a Design Research Methodology. Springer, London, 2009 – ISBN: 978-1-84882-587-1, DOI: 10.1007/978-1-84882-587-1

[BCK19] BAMBERG, G.; COENENBERG, A. G.; KRAPP, M.: Betriebswirtschaftliche Entscheidungslehre. Verlag Franz Vahlen, 16. Auflage, 2019 – ISBN: 978-3-8006-5884-8, DOI: 10.15358/9783800658855

[Ber12] BERGSMANN, S.: End-to-End-Geschäftsprozessmanagement – Organisationselement – Integrationsinstrument – Managementansatz. Springer, 2012, DOI: https://doi.org/10.1007/978-3-7091-0840-6

[BFP96] BÖHM, R.; FUCHS, E.; PACHER, G.: System-Entwicklung in der Wirtschaftsinformatik. vdf Hochschulverlag der ETH Zürich, 4. Auflage, 1996

[BGW18] BRACHT, U.; GECKLER, D.; WENZEL, S.: Digitale Fabrik – Methoden und Praxisbeispiele. Springer Vieweg, 2018 – ISBN: 978-3-662-55782-2, DOI: 10.1007/978-3-662-55783-9

[BH98] BAAKE, U. F.; HAUSSMANN, D. E.: Management of Concurrent Engineering Processes in international product development. In: Jacucci, G.; Olling, G. J.; Preiss, K.; Wozny, M. J.: Globalization of Manufacturing in the Digital Communications Era of the 21st Century – Innovation, Agility, and the Virtual Enterprise. Springer US, Boston, MA, 1998, pp. 1–12 – ISBN: 978-0-387-35351-7, DOI: 10.1007/978-0-387-35351-7_1

[BHS15] BAUER, W.; HERKOMMER, O.; SCHLUND, S.: Die Digitalisierung der Wertschöpfung kommt in deutschen Unternehmen an – Industrie 4.0 wird unsere Arbeit verändern. ZWF – Zeitschrift für wirtschaftlichen Fabrikbetrieb 110 (1-2), 2015, S. 68–73

[BKL+09] BALZERT, H.; KOSCHKE, R.; LÄMMEL, U.; LIGGESMEYER, P.; QUANTE, J.: Lehrbuch der Softwaretechnik – Basiskonzepte und Requirements Engineering. Spektrum Akademischer Verlag, 3. Auflage, 2009

[BL12] BLOHM, H.; LÜDER Klaus und Schaefer, C.: Investition – Schwachstellenanalyse des Investitionsbereichs und Investitionsrechnung. Verlag Franz Vahlen, 10. Auflage, 2012 – ISBN: 978-3-8006-3937-3, DOI: 10.15358/9783800639380_1

[BNM14] BERNUS, P.; NORAN, O.; MOLINA, A.: Enterprise Architecture – Twenty Years of the GERAM Framework. IFAC Proceedings Volumes 47 (3), Elsevier BV, 2014, pp. 3300–3308

[BPV12] BECKER, J.; PROBANDT, W.; VERING, O.: Grundsätze ordnungsmäßiger Modellierung – Konzeption und Praxisbeispiel für ein effizientes Prozessmanagement. Springer, 2012, DOI: https://doi.org/10.1007/978-3-642-30412-5

[BR16] BOSCHERT, S.; ROSEN, R.: Digital Twin – The Simulation Aspect. In: Hehenberger, P.; Bradley, D.: Mechatronic Futures – Challenges and Solutions for Mechatronic Systems and their Designers. Springer International, 2016, pp. 59–74 – ISBN: 978-3-319-32156-1, DOI: 10.1007/978-3-319-32156-1_5

[Bra05] BRAUN, T. E.: Methodische Unterstützung der strategischen Produktplanung in einem mittelständisch geprägten Umfeld. Dissertation, Fakultät für Maschinenwesen, 2005

[Bra13] BRAUN, A.: Modellbasierte Unterstützung der Produktentwicklung – Potentiale der Modellierung von Produktentstehungsprozessen am Beispiel des integrierten Produktentstehungsmodells (iPeM). Dissertation, KIT – Institut für Produktentwicklung (IPEK), 2013, DOI: DOI:10.5445/IR/1000040221

[Bro10] BROY, M. (Hrsg.): Cyber-Physical Systems – Innovation Durch Software-Intensive Eingebettete Systeme. Springer, Berlin, Heidelberg, 2010 – ISBN: 978-3-642-14498-1, DOI: 10.1007/978-3-642-14901-6

[BRS95] BECKER, J.; ROSEMANN, M.; SCHÜTTE, R.: Grundsätze ordnungsmäßiger Modellierung. Wirtschaftsinformatik 37 (5), 1995, S. 435–445

[Bru15] BRUNELLI, M.: Introduction to the Analytic Hierarchy Process. Springer, 2015 – ISBN: 978-3-319-12501-5, DOI: 10.1007/978-3-319-12502-2

[BZ14] BERG, C.; ZIMMERMANN, W.: Evaluierung von Möglichkeiten zur Implementierung von Semantischen Analysen für Domänenspezifische Sprachen. In: Schmid, K.; Böhm, W.; Heinrich, R. (Hrsg.), Software Engineering Workshops 2014, 25.-26. Februar 2014, Kiel, 2014, S. 111–128

[CJS98] CHOUIKHA, M.; JANHSEN, A.; SCHNIEDER, E.: Klassifikation und Bewertung von Beschreibungsmitteln für die Automatisierungstechnik. at – Automatisierungstechnik 46 (12), 1998, S. 582–591

[Com94] COMERFORD, R.: Mecha...what? IEEE Spectrum 31 (8), Aug. 1994, pp. 46–49

[Czi19] CZICHOS, H.: Mechatronik – Grundlagen und Anwendungen technischer Systeme. Springer Vieweg, 4. Auflage, 2019 – ISBN: 978-3-658-26293-8, DOI: 10.1007/978-3-658-26294-5

[DAR+21] DUMITRESCU, R.; ALBERS, A.; RIEDEL, O.; STARK, R.; GAUSEMEIER, J.: Engineering in Deutschland – Status quo in Wirtschaft und Wissenschaft – Ein Beitrag zum Advanced Systems Engineering, Paderborn, 2021, Unter: https://www.advanced-systems-engineering.de/documents/20210602_ASE_Engineering_in_Deutschland.pdf

[DB11] DRATH, R.; BARTH, M.: Concept for interoperability between independent engineering tools of heterogeneous disciplines. In: Mammeri, Z.; Sabatier (Eds.), Proceedings of the 2011 IEEE 16th Conference on Emerging Technologies and Factory Automation ETFA2011, September 5-9, 2001, Toulouse, France, IEEE, Sept. 2011

[DDF+14] DELLNITZ, M.; DUMITRESCU, R.; FLASSKAMP, K.; GAUSEMEIER, J.; HARTMANN, P.; IWANEK, P.; KORF, S.; KRÜGER, M.; OBER-BLÖBAUM, S.; PORRMANN, M.; PRIESTERJAHN, C.; STAHL, K.; TRÄCHTLER, A.; VASSHOLZ, M.: The Paradigm of Self-optimization. In: Jürgen Gausemeier; Rammig, F. J.; Wilhelm Schäfer: Design Methodology for Intelligent Technical Systems – Develop Intelligent Technical Systems of the Future. Springer, Berlin, 2014, pp. 1–25 – ISBN: 978-3-642-45434-9, DOI: 10.1007/978-3-642-45435-6_1

[DDG+14] DOROCIAK, R.; DUMITRESCU, R.; GAUSEMEIER, J.; IWANEK, P.: Specification Technique CONSENS for the Description of Self-optimizing Systems. In: Jürgen Gausemeier; Rammig, F. J.; Wilhelm Schäfer: Design Methodology for Intelligent Technical Systems – Develop Intelligent Technical Systems of the Future. Springer, Berlin, 2014, pp. 119–127 – ISBN: 978-3-642-45434-9, DOI: 10.1007/978-3-642-45435-6_4

[DGO+14] DZIWOK, S.; GAUSEMEIER, J.; OESTERSÖTEBIER, F.; POHLMANN, U.; RIEKE, J.; SCHÄFER, W.; SCHIERBAUM, T.; TRÄCHTLER, A.: Modellbasierter Entwurf mechatronischer Systeme. In: Gausemeier, J.; Trächtler, A.; Schäfer, W.: Semantische Technologien im Entwurf mechatronischer Systeme. Carl Hanser, Juni 2014, – ISBN: 978-3-446-43630-5, DOI: 10.3139/9783446438453

[DR13] DRESCHER, B.; REINHART, G.: Unternehmensspezifische Zusammenstellung und Bewertung digitaler Werkzeugketten zur Unterstützung mechatronischer Anlagenentstehungsprozesse. In: Maurer, M.; Schulze, S.-O. (Hrsg.), Tag des Systems Engineering – Stuttgart, 6.-8. November 2013, Carl Hanser, 2013, S. 229–238

[Dum10] DUMITRESCU, R.: Entwicklungssystematik zur Integration kognitiver Funktionen in fortgeschrittene mechatronische Systeme. Fakultät Maschinenbau. Dissertation, Universität Paderborn, Paderborn, 2010

[Ebe19] EBERT, C.: Systematisches Requirements Engineering – Anforderungen ermitteln, dokumentieren, analysieren und verwalten. dpunkt.verlag, Heidelberg, 6. Auflage, 2019 – ISBN: 978-3-86490-562-9

[Ech17] ECHTERHOFF, O.: Systematik zur Erarbeitung modellbasierter Entwicklungsaufträge. Dissertation, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 362, Jan. 2017

[Eig13] EIGNER, M.: Modellbasierte Virtuelle Produktentwicklung auf einer Plattform für System Lifecycle Management. In: Sendler, U.: Industrie 4.0 – Beherrschung der industriellen Komplexität mit SysLM. Springer, 2013, S. 91–110 – ISBN: 978-3-642-36917-9, DOI: 10.1007/978-3-642-36917-9_6

[Eig14a] EIGNER, M.: Product Lifecycle Management (PLM). In: Eigner, M.; Roubanov, D.; Zafirov, R.: Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 267–300 – ISBN: 978-3-662-43816-9, DOI: 10.1007/978-3-662-43816-9_12

[Eig14b] EIGNER, M.: Überblick – Disziplin-spezifische und -übergreifende Vorgehensmodelle. In: Modellbasierte Virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 15–52, DOI: 10.1007/978-3-662-45874-7_2

[Eig16] EIGNER, M.: IT-Lösungen für den Produktentwicklungsprozess. In: Spath, D.; Westkämper, E.: Handbuch Unternehmensorganisation. Springer Vieweg, Berlin, Heidelberg, 2016, S. 1–19 – ISBN: 978-3-642-45370-0, DOI: 10.1007/978-3-642-45370-0_24-1

[EMA+16] EIGNER, M.; MUGGEO, C.; APOSTOLOV, H.; SCHÄFER, P.: Kern des System Lifecycle Management – Im Kontext von Industrial Internet mit Industrie 4.0 und Internet der Dinge und Dienste. ZWF – Zeitschrift für wirtschaftlichen Fabrikbetrieb 111 (1-2), Hanser Verlag, Dez. 2016, S. 63–68

[ERZ14] EIGNER, M.; ROUBANOV, D.; ZAFIROV, R. (Hrsg.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin, 2014 – ISBN: 978-3-662-43815-2, DOI: 10.1007/978-3-662-43816-9

[ES09] EIGNER, M.; STELZER, R.: Product Lifecycle Management – Ein Leitfaden für Product Development und Life Cycle Management. Springer, 2. Auflage, 2009 – ISBN: 978-3-540-44373-5, DOI: https://doi.org/10.1007/b93672

[Fah95] FAHRWINKEL, U.: Methoden zur Modellierung und Analyse von Geschäftsprozessen zur Unterstützung des Business Process Reengineering, Fakultät für Wirtschaftswissenschaften, Universität-Gesamthochschule Paderborn, 1995, – ISBN: 978-3-931466-00-8

[FBK+14] FRIEDENTHAL, S.; BEIHOFF, B.; KEMP, D.; NICHOLS, D.; OSTER, C.; PAREDIS, C.; STOEWER, H.; WADE, J.: A World in Motion – Systems Engineering Vision 2025, 2014, Unter: https://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf?sfvrsn=b69eb4c6_4

[FDN+12] FISCHER, J.; DANGELMAIER, W.; NASTANSKY, L.; SUHL, L.: Bausteine der Wirtschaftsinformatik – Grundlagen und Anwendungen. Erich Schmidt Verlag, 5. Auflage, 2012 – ISBN: 978-3-503-13865-4

[FG13] FELDHUSEN, J.; GROTE, K.-H. (Hrsg.): Pahl/Beitz Konstruktionslehre – Methoden und Anwendung erfolgreicher Produktentwicklung. Springer Vieweg, 8. Auflage, 2013 – ISBN: 978-3-642-29568-3, DOI: https://doi.org/10.1007/978-3-642-29569-0

[FGS08] FRICK, D.; GADATSCH, A.; SCHÄFFER-KÜLZ, U. G.: Grundkurs SAP ERP – Geschäftsprozessorientierte Einführung mit durchgehendem Fallbeispiel. Vieweg & Sohn Verlag, Wiesbaden, 1. Auflage, 2008 – ISBN: 978-3-8348-0361-0, DOI: https://doi.org/10.1007/978-3-8348-9264-5

[FHF+17] FRIEDLAND, B.; HERROLD, J.; FERGUSON, G.; MALONE, R.: Conducting a Model Based Systems Engineering Tool Trade Study Using a Systems Engineering Approach. INCOSE International Symposium 27 (1), 2017, pp. 1087–1099

[FL12] FISCHER, R.; LINSE, H.: Elektrotechnik für Maschinenbauer – mit Elektronik, elektrischer Messtechnik, elektrischen Antrieben und Steuerungstechnik. Springer Vieweg, 14. Auflage, 2012 – ISBN: 978-3-8348-1374-9, DOI: https://doi.org/10.1007/978-3-8348-8304-9

[FLL+11] FRIEDEWALD, A.; LÖDDING, H.; LUKAS, U. F. v.; MESING, B.; ROTH, M.; SCHLEUSENER, S.; TITOV, F.: Benchmark neutraler Formate für den prozessübergreifenden Datenaustausch im Schiffbau, Aug. 2011

[FMH16] FRIEDLAND, B.; MALONE, R.; HERROLD, J.: Systems Engineering a Model Based Systems Engineering Tool Suite – The Boeing Approach. INCOSE International Symposium 26 (1), 2016, pp. 386–398

[FMS12] FRIEDENTHAL, S.; MOORE, A.; STEINER, R.: A Practical Guide to SysML – The Systems Modeling Language. Morgan Kaufmann, 2nd Edition, 2012 – ISBN: 978-0-12-385206-9, DOI: https://doi.org/10.1016/C2010-0-66331-0

[FNF+14] FISHER, A.; NOLAN, M.; FRIEDENTHAL, S.; LOEFFLER, M.; SAMPSON, M.; BAJAJ, M.; VANZANDT, L.; HOVEY, K.; PALMER, J.; HART, L.: Model Lifecycle Management for MBSE. In: INCOSE International Symposium, Las Vegas, NV, June 30-July 3, 2014, Wiley, July 2014, pp. 207–229

[FOS+18] FLEISCHMANN, A.; OPPL, S.; SCHMIDT, W.; STARY, C.: Ganzheitliche Digitalisierung von Prozessen – Perspektivenwechsel – Design Thinking – Wertegeleitete Interaktion. Springer, 2018, DOI: https://doi.org/10.1007/978-3-658-22648-0

[FR16] FREUND, J.; RÜCKER, B.: Praxishandbuch BPMN – Mit Einführung in CMMN und DMN. Carl Hanser, München, 5. Auflage, 2016 – ISBN: 978-3-446-45054-7, DOI: 10.3139/9783446450783

[Fra06] FRANK, U.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme, Band 175, 2006

[Fra21] FRANK, M.: Systematik zur Planung des organisationalen Wandels zum Smart Service-Anbieter. Fakultät Maschinenbau. Dissertation, Universität Paderborn, Paderborn, 2021, DOI: 10.17619/UNIPB/1-1130

[Gad17] GADATSCH, A.: Grundkurs Geschäftsprozess-Management – Analyse, Modellierung, Optimierung und Controlling von Prozessen. Springer Vieweg, Wiesbaden, 8. Auflage, 2017 – ISBN: 978-3-658-17178-0, DOI: 10.1007/978-3-658-17179-7

[GDE+18] GAUSEMEIER, J.; DUMITRESCU, R.; ECHTERFELD, J.; PFÄNDER, T.; STEFFEN, D.; THIELEMANN, F.: Innovationen für die Märkte von morgen – Strategische Planung von Produkten, Dienstleistungen und Geschäftsmodellen. Carl Hanser, München, 2018 – ISBN: 978-3-446-42972-7, DOI: 10.3139/9783446429727

[GDS+13] GAUSEMEIER, J.; DUMITRESCU, R.; STEFFEN, D.; CZAJA, A.; WIEDERKEHR, O.; TSCHIRNER, C.: Systems Engineering in der industriellen Praxis, Paderborn, 2013, Unter: https://www.iem.fraunhofer.de/content/dam/iem/de/documents/Studie%20Systems%20Engineering_deutsch.pdf

[Ger16] GERHARD, D.: Daten- und Informationsmanagement PDM/PLM. In: Lindemann, U.: Handbuch Produktentwicklung. Carl Hanser Verlag GmbH & Co. KG, Dez. 2016, S. 215–246 – ISBN: 978-3-446-44518-5, DOI: 10.3139/9783446445819.009

[Ger17] GERLMAIER, A.: Organisationale Gestaltungskompetenz im Betrieb – Ein (unterschätzter) Mediator des Zusammenhangs von psychischer Belastung und Beanspruchung. Zeitschrift für Arbeitswissenschaft 72 (2), Springer Nature, Dez. 2017, S. 130–136

[GFD+08] GAUSEMEIER, J.; FRANK, U.; DONOTH, J.; KAHL, S.: Spezifikationstechnik zur Beschreibung der Prinziplösung selbstoptimierender Systeme des Maschinenbaus (Teil 2). Konstruktion 9/2008 (9), S. 91–108

[GLL12] GAUSEMEIER, J.; LANZA, G.; LINDEMANN, U.: Produkte und Produktionssysteme integrativ konzipieren - Modellbildung und Analyse in der frühen Phase der Produktentstehung. Carl Hanser, 2012

[Goe18] GOEPEL, K. D.: Implementation of an Online Software Tool for the Analytic Hierarchy Process (AHP-OS). International Journal of the Analytic Hierarchy Process 10 (3), Creative Decisions Foundation, Dec. 2018

[Goe19] GOEPEL, K. D.: AHP Online System – AHP-OS, Abgerufen am 17.05.2020, 2019, Unter: https://bpmsg.com/ahp/ahp-calc.php (besucht am 17. 05. 2020)

[Göt14] GÖTZE, U.: Investitionsrechnung – Modelle und Analysen zur Beurteilung von Investitionsvorhaben. Springer Gabler, 7. Auflage, 2014 – ISBN: 978-3-642-54621-1, DOI: 10.1007/978-3-642-54622-8

[GP14] GAUSEMEIER, J.; PLASS, C.: Zukunftsorientierte Unternehmensgestaltung. Carl Hanser, Jan. 2014, DOI: 10.3139/9783446438422

[GR96] GALLAUGHER, J. M.; RAMANATHAN, S. C.: Choosing a Client/Server Architecture – A Comparison of Two-and Three-Tier Systems. Information Systems Management 13 (2), Jan. 1996, pp. 7–13

[Gra13] GRANER, M.: Der Einsatz von Methoden in Produktentwicklungsprojekten – Eine empirische Untersuchung der Rahmenbedingungen und Auswirkungen. Springer Gabler, 2013 – ISBN: 978-3-658-01277-9, DOI: https://doi.org/10.1007/978-3-658-01278-6

[Gre03] GREULE, F.: Die Integration des Computers in den Unterricht der Volksschule. Beiträge zur Lehrerinnen- und Lehrerbildung 21 (1), 2003, S. 89–92

[Gri10] GRIMM, R.: Der operative IT-Strategie-Ansatz. In: Keuper, F.; Schomann, M.; Zimmermann, K.: Innovatives IT-Management – Management von IT und IT-gestütztes Management. Gabler, 2010, S. 71–97 – ISBN: 978-3-8349-8803-4, DOI: 10.1007/978-3-8349-8803-4_4

[GZ14] GILZ, T.; ZAFIROV, R.: Modellbildung und erste Simulation. In: Eigner, M.; Roubanov, D.; Zafirov, R.: Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 97–113 – ISBN: 978-3-662-43816-9, DOI: 10.1007/978-3-662-43816-9_5

[Han16] HANSCHKE, I.: Enterprise Architecture Management – einfach und effektiv – Ein praktischer Leitfaden für die Einführung von EAM. Carl Hanser, München, 2., überarbeitete Auflage, 2016 – ISBN: 978-3-446-44724-0, DOI: 10.3139/9783446449350

[Har49] HARTMANN, N.: Der Aufbau der realen Welt – Grundriß der allgemeinen Kategorienlehre. Hain, 2. Auflage, 1949

[Heh11] HEHENBERGER, P.: Computerunterstützte Fertigung – Eine kompakte Einführung. Springer, Berlin, Heidelberg, 2011 – ISBN: 978-3-642-13474-6, DOI: 10.1007/978-3-642-13475-3

[HGG16] HANSCHKE, I.; GIESINGER, G.; GOETZE, D.: Business Analyse – einfach und effektiv – Geschäftsanforderungen verstehen und in IT-Lösungen umsetzen. Carl Hanser, München, 2. Auflage, 2016 – ISBN: 978-3-446-44345-7, DOI: 10.3139/9783446444218

[HH11] HUBER, M.; HUBER, G.: Prozess- und Projektmanagement für ITIL – Nutzen Sie ITIL optimal. Vieweg+Teubner, Wiesbaden, 2011 – ISBN: 978-3-8348-1482-1, DOI: 10.1007/978-3-8348-8195-3

[HH18] HOFFMANN, J.; HEIMES, P.: Informationssystem-Architekturen produzierender Unternehmen für die Digitalisierung gestalten. HMD Praxis der Wirtschaftsinformatik 55 (5), S. 984–1005

[HHG13] HAHN, A.; HÄUSLER, S.; GROSSE AUSTING, S.: Quantitatives Entwicklungsmanagement – Modellbasierte Analyse von Produktentwicklungsprozessen. Springer Vieweg, 2013 – ISBN: 978-3-642-34509-8, DOI: https://doi.org/10.1007/978-3-642-34510-4

[Hit07] HITCHINS, D. K.: Systems engineering – a 21st century systems methodology. Wiley, 2007

[HIW+18] HUTH, T.; INKERMANN, D.; WILMS, R.; VIETOR, T.: Model-based Process Engineering – An approach to integrated product system and process modelling, INCOSE/GfSE, 2018

[HJD11] HULL, E.; JACKSON, K.; DICK, J.: Requirements Engineering. Springer, 3rd Edition, 2011 – ISBN: 978-1-84996-404-3

[HK16] HELMS, B.; KISSEL, M.: Engineering Intelligence? – Von der graphenbasierten Modellierung zur wissensbasierten Datenanalyse. In: Lindemann, U.: Handbuch Produktentwicklung. Carl Hanser, 2016, S. 979–1012 – ISBN: 978-3-446-44518-5, DOI: 10.3139/9783446445819.035

[HN12] HANSMANN, H.; NEUMANN, S.: Prozessorientierte Einführung von ERP-Systemen. In: Becker, J.; Kugeler, M.; Rosemann, M.: Prozessmanagement – Ein Leitfaden zur prozessorientierten Organisationsgestaltung. Springer Gabler, 7. Auflage, 2012, S. 329–366 – ISBN: 978-3-642-33843-4, DOI: 10.1007/978-3-642-33844-1_10

[Hor01] HORVÁTH, I.: A contemporary survey of scientific research into engineering design. In: International Conference on Engineering Design ICED 01, August 21-23, 2001, Glasgow, Scotland, 2001, S. 13–20

[HTF96] HARASHIMA, F.; TOMIZUKA, M.; FUKUDA, T.: Mechatronics – What Is It, Why, and How? An editorial. IEEE/ASME Transactions on Mechatronics (IEEE/ASME Transactions on Mechatronics) 1 (1), 1996, pp. 1–4

[HWF+18] HABERFELLNER, R.; WECK, O. de; FRICKE, E.; VÖSSNER, S.: Systems Engineering – Grundlagen und Anwendung. Orell Füssli, 14. Auflage, 2018

[HZ11] HERDEN, S.; ZENNER, U.: Klassifikation von Enterprise-Architecture-Management-Frameworks – Eine Literaturanalyse, Technischer Bericht, 2011, Unter: https://www.inf.ovgu.de/inf_media/downloads/forschung/technical_reports_und_preprints/2011/TechReport07-p-3592.pdf

[IAE19-ol] IAENG INTERNATIONAL ASSOCIATION OF ENGINEERS (Hrsg.): About IAENG. Unter: http://www.iaeng.org/about_IAENG.html, 2019

[Ise08] ISERMANN, R.: Mechatronische Systeme – Grundlagen. Springer, Berlin, Heidelberg, 2. vollständig neu bearbeitete Auflage, 2008 – ISBN: 978-3-540-32336-5, DOI: 10.1007/978-3-540-32512-3

[its19] IT'S OWL CLUSTERMANAGEMENT (Hrsg.): KI-Plattform für die Produkte von morgen, 2019, Unter: https://www.its-owl.de/newsroom/news/news/news/ki-plattform-fuer-die-produkte-von-morgen/ (besucht am 24. 12. 2019)

[Jop21] JOPPEN, R.: Systematik zur Bewertung von Industrie 4.0 Anwendungen in der Produktion. Fakultät Maschinenbau. Dissertation, Universität Paderborn, Paderborn, 2021, DOI: 10.17619/UNIPB/1-1220

[Kai14] KAISER, L.: Rahmenwerk zur Modellierung einer plausiblen Systemstruktur mechatronischer Systeme. Dissertation, Verlagsschriftenreihe des Heinz Nixdorf Instituts, Paderborn, Band 327, 2014

[Kel17] KELLER, W.: IT-Unternehmensarchitektur – Von der Geschäftsstrategie zur optimalen IT-Unterstützung. dpunkt.Verlag, 3. Auflage, 2017 – ISBN: 978-3-86490-406-6

[KK02] KARAGIANNIS, D.; KÜHN, H.: Metamodelling Platforms. In: Bauknecht, K.; Tjoa, A. M.; Quirchmayr, G.: E-Commerce and Web Technologies – Third International Conference, EC-Web 2002 Aix-en-Provence, France, September 2–6, 2002 Proceedings. Springer, 2002, pp. 182–182, DOI: 10.1007/3-540-45705-4_19

[KLW12] KORFF, A.; LAMM, J. G.; WEILKIENS, T.: Werkzeuge für den Schmied funktionaler Architekturen. In: Maurer, M.; Schulze, S.-O. (Hrsg.), Tag des Systems Engineering 2011 - Hamburg, 9.-11. November 2011, Carl Hanser, 2012

[Köc04] KÖCKERLING, M.: Methodische Entwicklung und Optimierung der Wirkstruktur mechatronischer Produkte. Dissertation, Fakultät für Maschinenbau, Universität Paderborn, 2004, – ISBN: 3-935433-52-2

[Krc15] KRCMAR, H.: Informationsmanagement. Springer, 6. Auflage, 2015 – ISBN: 978-3-662-45862-4, DOI: 10.1007/978-3-662-45863-1

[Kro12] KROGSTIE, J.: Model-Based Development and Evolution of Information Systems – A Quality Approach. Springer, 2012 – ISBN: 978-1-4471-2935-6, DOI: 10.1007/978-1-4471-2936-3

[Kur08] KURBEL, K. E.: The Making of Information Systems – Software Engineering and Management in a Globalized World. Springer, 2008 – ISBN: 978-3-540-79260-4, DOI: 10.1007/978-3-540-79261-1

[KV08] KÄÄRIÄINEN, J.; VÄLIMÄKI, A.: Impact of Application Lifecycle Management – A Case Study. In: Mertins, K.; Ruggaber, R.; Popplewell, K.; Xu, X. (Eds.), Enterprise Interoperability III, Springer London, 2008, pp. 55–67

[Lan09] LANNINGER, V.: Prozessmodell zur Auswahl Betrieblicher Standardanwendungssoftware für KMU. Josef Eul Verlag, 2009 – ISBN: 978-3-89936-870-3

[Lan17] LANKHORST, M.: Enterprise Architecture at Work – Modelling, Communication and Analysis. Springer, 4th Edition, 2017 – ISBN: 978-3-662-53932-3, DOI: https://doi.org/10.1007/978-3-662-53932-3

[LB15] LEMKE, C.; BRENNER, W.: Einführung in die Wirtschaftsinformatik – Band 1 – Verstehen des digitalen Zeitalters. Springer Gabler, 2015 – ISBN: 978-3-662-44064-3, DOI: 10.1007/978-3-662-44065-0

[Lei15] LEIMEISTER, J. M.: Einführung in die Wirtschaftsinformatik. Springer Gabler, 12. Auflage, 2015 – ISBN: 978-3-540-77846-2, DOI: https://doi.org/10.1007/978-3-540-77847-9

[LG20] LUNTOVSKYY, A.; GÜTTER, D.: Moderne Rechnernetze – Protokolle, Standards und Apps in kombinierten drahtgebundenen, mobilen und drahtlosen Netzwerken. Springer Vieweg, 2020 – ISBN: 978-3-658-25616-6, DOI: 10.1007/978-3-658-25617-3

[LHS08] LAUGWITZ, B.; HELD, T.; SCHREPP, M.: Construction and Evaluation of a User Experience Questionnaire. In: Holzinger, A.: HCI and Usability for Education and Work – 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2008, Graz, Austria, November 20-21, 2008. Springer, 2008, pp. 63–76 – ISBN: 978-3-540-89349-3, DOI: 10.1007/978-3-540-89350-9_6

[Lin09] LINDEMANN, U.: Methodische Entwicklung technischer Produkte – Methoden flexibel und situationsgerecht anwenden. Springer, 3. Auflage, 2009 – ISBN: 978-3-642-01422-2, DOI: https://doi.org/10.1007/978-3-642-01423-9

[LLS16] LAUDON, K. C.; LAUDON, J. P.; SCHODER, D.: Wirtschaftsinformatik. Pearson Studium, 3. Auflage, 2016 – ISBN: 978-3-8689-4269-9

[Lon12] LONG, J. O.: ITIL 2011 At a Glance. Springer, New York, NY, 2012 – ISBN: 978-1-4614-3896-0, DOI: 10.1007/978-1-4614-3897-7

[Lu19] LU, J.: A Framework for Cyber-physical System Tool-chain Development – A Service-oriented and Model-based Systems Engineering Approach. PhD thesis, 2019

[LW10] LAMM, J. G.; WEILKIENS, T.: Funktionale Architekturen in SysML. In: Maurer, M.; Schulze, S.-O. (Hrsg.), Tag des Systems Engineering 2010 – München, Freising, 10.-12. November 2010, Carl Hanser, 2010

[LW14] LAMM, J. G.; WEILKIENS, T.: Method for Deriving Functional Architectures from Use Cases. Systems Engineering 17 (2), 2014, pp. 225–236

[Mat11] MATTHES, D.: Enterprise Architecture Frameworks Kompendium. Springer, 2011 – ISBN: 978-3-642-12954-4, DOI: 10.1007/978-3-642-12955-1

[MBK+17] MERTENS, P.; BODENDORF, F.; KÖNIG, W.; SCHUMANN, M.; HESS, T.; BUXMANN, P.: Grundzüge der Wirtschaftsinformatik. Springer, 9. Auflage, 2017 – ISBN: 978-3-540-23411-1, DOI: https://doi.org/10.1007/b138341

[MBK+19] MEFFERT, H.; BURMANN, C.; KIRCHGEORG, M.; EISENBEISS, M.: Marketing – Grundlagen marktorientierter Unternehmensführung. Springer Gabler, 13. Auflage, 2019 – ISBN: 978-3-658-21195-0, DOI: 10.1007/978-3-658-21196-7

[MFH15] MEYER, J.; FOCKEL, M.; HOLTMANN, J.: Systementwurf unter Einbeziehung funktionaler Sicherheit bei automobilen Steuergeräten. In: Schulze, S.-O.; Muggeo, C. (Hrsg.), Tag des Systems Engineering 2015 – Ulm, 11.-13. November 2015, Carl Hanser, 2015

[MFL19] MARTIN; FAIRLEY, D.; LAWSON, B.: Enterprise Systems Engineering – Systems Engineering Body of Knowledge (SEBoK), 2019, Unter: https://www.sebokwiki.org/wiki/Enterprise_Systems_Engineering (besucht am 02. 12. 2019)

[MK13] MÜHLBACHER, A. C.; KACZYNSKI, A.: Der Analytic Hierarchy Process (AHP) – Eine Methode zur Entscheidungsunterstützung im Gesundheitswesen. PharmacoEconomics German Research Articles 11 (2), Springer, Okt. 2013, S. 119–132

[Moo09] MOODY, D.: The "Physics" of Notations – Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering 35 (6), pp. 756–779

[MP18] MU, E.; PEREYRA-ROJAS, M.: Practical Decision Making using Super Decisions v3 – An Introduction to the Analytic Hierarchy Process. Springer, 2018 – ISBN: 978-3-319-68368-3, DOI: 10.1007/978-3-319-68369-0

[MRE14] MENGES, M.; ROUBANOV, D.; ERNST, J.: Produktionsplanung und -steuerung (PPS). In: Modellbasierte Virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 301–325 – ISBN: 978-3-662-43815-2, DOI: 10.1007/978-3-662-45874-7_13

[Mül19] MÜLLER, D.: Investitionsrechnung und Investitionscontrolling. Springer Gabler, 2. Auflage, 2019 – ISBN: 978-3-662-57608-3, DOI: 10.1007/978-3-662-57609-0

[Nus01] NUSEIBEH, B.: Weaving together requirements and architectures. Computer 34 (3), Institute of Electrical and Electronics Engineers (IEEE), Mar. 2001, pp. 115–119

[OB09] OESTEREICH, B.; BREMER, S.: Analyse und Design mit der UML 2.3 – Objektorientierte Softwareentwicklung. Oldenbourg Wissenschaftsverlag, 9. Auflage, 2009 – ISBN: 978-3-486-58855-2

[OB18] OTTO, J.; BARTELS, N.: Integration von FM-Prozessdaten in ein digitales Gebäudemodell. Bautechnik 95 (12), Ernst & Sohn Verlag, Okt. 2018, S. 823–831

[OMG08-ol] OBJECT MANAGEMENT GROUP, O.: Software & Systems Process Engineering Metamodel Specification 2.0. Unter: https://www.omg.org/spec/SPEM/2.0, Apr. 2008

[OMG14-ol] OBJECT MANAGEMENT GROUP, O.: Business Process Model and Notation Specification 2.0.2. Unter: https://www.omg.org/spec/BPMN/2.0.2/, Jan. 2014

[OMG16a-ol] OBJECT MANAGEMENT GROUP, O.: Meta Object Facility Core Specification 2.5.1. Unter: https://www.omg.org/spec/MOF/2.5.1, Okt. 2016

[OMG16b-ol] OBJECT MANAGEMENT GROUP, O.: Requirements Interchange Format Specification 1.2. Unter: https://www.omg.org/spec/ReqIF/1.2, Dez. 2016

[OMG17-ol] OBJECT MANAGEMENT GROUP, O.: Unified Modeling Language Specification 2.5.1. Unter: https://www.omg.org/spec/SysML/1.6, Dez. 2017

[OMG19-ol] OBJECT MANAGEMENT GROUP, O.: Systems Modeling Language Specification 1.6. Unter: https://www.omg.org/spec/SysML/1.6, Dez. 2019

[ÖO10] ÖSTERLE, H.; OTTO, B.: Konsortialforschung. Wirtschaftsinformatik 52 (5), Aug. 2010, S. 273–285

[Pat82] PATZAK, G.: Systemtechnik – Planung komplexer innovativer Systeme – Grundlagen, Methoden, Techniken. Springer, 1982 – ISBN: 978-3-540-11783-4, DOI: 10.1007/978-3-642-81893-6

[PM06] PETRASCH, R.; MEIMBERG, O.: Model driven architecture – Eine praxisorientierte Einführung in die MDA. dpunkt.verlag, 2006

[Poh08] POHL, K.: Requirements Engineering – Grundlagen, Prinzipien,Techniken. dpunkt.verlag, 2. Auflage, 2008 – ISBN: 978-3-89864-550-8

[PR15] POHL, K.; RUPP, C.: Basiswissen Requirements Engineering – Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level. dpunkt.verlag, 4. Auflage, 2015 – ISBN: 978-3-86490-283-3

[Rau07] RAU, K.-H.: Objektorientierte Systementwicklung – Vom Geschäftsprozess zum Java-Programm. Vieweg, 2007 – ISBN: 978-3-8348-0245-3, DOI: 10.1007/978-3-8348-9174-7

[Rei07] REISCH, M.: Elektronische Bauelemente – Funktion, Grundschaltungen, Modellierung mit SPICE. Springer, Berlin, Heidelberg, 2., vollständig neu bearbeitete Auflage, 2007 – ISBN: 978-3-540-34014-0, DOI: 10.1007/978-3-540-34015-7

[RFB12] RAMOS, A. L.; FERREIRA, J. V.; BARCELÓ, J.: Model-Based Systems Engineering – An Emerging Approach for Modern Systems. IEEE Trans"-actions On Systems, Man, And Cybernetics – Part C: Applications And Reviews 42 (1), Jan. 2012, pp. 101–111

[RNW16] RIEG, F.; NÜTZEL, F.; WEHMANN, C.: Numerische Simulationsverfahren. In: Lindemann, U.: Handbuch Produktentwicklung. Carl Hanser Verlag GmbH & Co. KG, Dez. 2016, S. 905–937 – ISBN: 978-3-446-44518-5, DOI: 10.3139/9783446445819.032

[Rod19] RODDECK, W.: Einführung in die Mechatronik. Springer Vieweg, Wiesbaden, 6. Auflage, 2019 – ISBN: 978-3-658-27774-1, DOI: 10.1007/978-3-658-27775-8

[Rop09] ROPOHL, G.: Allgemeine Technologie – eine Systemtheorie der Technik. Universitätsverlag Karlsruhe, Karlsruhe, 3., überarbeitete Auflage, 2009 – ISBN: 978-3-86644-374-7, DOI: http://dx.doi.org/10.5445/KSP/1000011529

[Rou14] ROUBANOV, D.: Produktmodelle und Simulation (CAE). In: Eigner, M.; Roubanov, D.; Zafirov, R.: Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 175–195 – ISBN: 978-3-662-43816-9, DOI: 10.1007/978-3-662-43816-9_9

[RQ12] RUPP, C.; QUEINS, S.: UML 2 glasklar – Praxiswissen für die UML-Modellierung. Carl Hanser, 4. Auflage, 2012 – ISBN: 978-3-446-43057-0, DOI: 10.3139/9783446431973

[RS14] RUPP, C.; DIE SOPHISTEN: Requirements-Engineering und Management – Aus der Praxis von klassisch bis agil. Carl Hanser, München, 6. Auflage, 2014 – ISBN: 978-3-446-43893-4, DOI: 10.3139/9783446443136

[RWZ+17] RYAN, M.; WHEATCRAFT, L.; ZINNI, R.; DICK, J.; BAKSA, K.: Guide for Writing Requirements, Apr. 2017

[Saa08] SAATY, T. L.: Decision making with the analytic hierarchy process. International Journal of Services Sciences 1 (1), Inderscience, 2008, p. 83

[Saa13] SAATY, T. L.: Analytic Hierarchy Process. In: Gass, S. I.; Fu, M. C.: Encyclopedia of Operations Research and Management Science. Springer, 2013, pp. 52–64 – ISBN: 978-1-4419-1153-7, DOI: 10.1007/978-1-4419-1153-7_31

[Saa77] SAATY, T. L.: A scaling method for priorities in hierarchical structures. Journal of Mathematical Psychology 15 (3), Elsevier, June 1977, pp. 234–281

[Sae05] SAENZ, O. A.: Framework for Enterprise Systems Engineering, Florida International University, 2005, DOI: 10.25148/etd.fi08081544

[Sar11] SARODNICK Florian; Brau, H.: Methoden der Usability-Evaluation – wissenschaftliche Grundlagen und praktische Anwendung. Verlag Hans Huber, 2. Auflage, 2011 – ISBN: 978-3-456-84883-9

[SB07] SEARLE, J.; BRENNAN, J.: General Interoperability Concepts. In: Integration of Modelling and Simulation. Research and Technology Organisation (RTO) of NATO, 2007, – ISBN: 978-92-837-0046-3, DOI: 10.14339/RTO-EN-MSG-067-03-pdf

[SBI+18] STÜTZEL, B.; BORCHARDT, L.; ILLA, T.; GERLING, C.: Studie Systems Engineering in Deutschland – Die deutsche Unternehmenslandschaft im Vergleich, GmbH, P., Unterföhring, 2018, Unter: http://www.prozesswerk.eu/site/assets/files/1179/se_studie_prozesswerk_doppelseiten.pdf

[SBM+12] SCHUH, G.; BROSZE, T.; MEIER, C. v.; SCHMIDT, C. v.; BAUHOFF, F.; SCHOTH, A.; RUNGE, S.; OEDEKOVEN, D. v.; BAUHOFF, F.; KOMPA, S.; SCHENK, M. v.; SCHÜRMEYER, M.: Gestaltung der innerbetrieblichen Produktionsplanung und -steuerung. In: Schuh, G.; Stich, V.: Produktionsplanung und -steuerung 1 – Evolution der PPS. Springer Vieweg, 4. Auflage, 2012, S. 297–470 – ISBN: 978-3-642-25422-2, DOI: 10.1007/978-3-642-25423-9_3

[SCC+09] SAENZ, O.; CHEN, S.; CENTENO, M.; GIACHETTI, R.: Defining Enterprise Systems Engineering. International Journal of Industrial and Systems Engineering 4 (5), Jan. 2009

[Sch01] SCHABACKER, M.: Bewertung der Nutzen neuer Technologien in der Produktentwicklung. Dissertation, 2001

[Sch09] SCHIRRA, R.: Entwicklung eines Systems zur immersiven Arbeitssystemgestaltung mit CAD-Schnittstelle, 2009, – ISBN: 978-3-939890-39-3, DOI: Schirra,Roman

[Sch13] SCHUH, G.: Lean Innovation. Springer Vieweg, 2013 – ISBN: 978-3-540-76914-9, DOI: 10.1007/978-3-540-76915-6

[Sch17] SCHÜTZ, A.: Komplexität von IT-Architekturen – Konzeptualisierung, Quantifizierung, Planung und Kontrolle. Dissertation, 2017, DOI: https://doi.org/10.1007/978-3-658-18226-7

[Sch18] SCHNEIDER, M.: Spezifikationstechnik zur Beschreibung und Analyse von Wertschöpfungssystemen. Fakultät Maschinenbau. Dissertation, Universität Paderborn, Paderborn, 2018

[Sch19] SCHREPP, M.: User Experience Questionnaire – Handbook, Version Version 8, Abgerufen am 16.5.2020, 2019, Unter: https://www.ueq-online.org/Material/Handbook.pdf (besucht am 16. 05. 2020)

[SD16] SPATH, D.; DANGELMAIER, M.: Produktentwicklung Quo Vadis. In: Lindemann, U.: Handbuch Produktentwicklung. Carl Hanser Verlag GmbH & Co. KG, Dez. 2016, S. 1–7 – ISBN: 978-3-446-44518-5, DOI: 10.3139/9783446445819.001

[SEL13] SPEICHER, S.; ERSCH, R.; LOEFFLER, M. (Hrsg.): Open Services for Lifecycle Collaboration Core Specification Version 2.0, OSLC Open Project, Mai 2013, Unter: https://archive.open-services.net/bin/view/Main/OslcCoreSpecification.html (besucht am 11. 01. 2020)

[SH93] SPEWAK, S. H.; HILL, S. C.: Enterprise Architecture Planning – Developing a Blueprint for Data, Applications, and Technology. QED Publishing, 1993 – ISBN: 9780894354366

[SHT14] SCHREPP, M.; HINDERKS, A.; THOMASCHEWSKI, J.: Applying the User Experience Questionnaire (UEQ) in Different Evaluation Scenarios. In: Marcus, A.: Design, User Experience, and Usability – Theories, Methods, and Tools for Designing the User Experience. Springer, 2014, pp. 383–392 – ISBN: 978-3-319-07667-6, DOI: 10.1007/978-3-319-07668-3_37

[Sie10] Teamcenter's Service Oriented Architecture – Integrating Teamcenter-managed product design and development with your mainstream business processes, 2010, Unter: https://www.plm.automation.siemens.com/fr_fr/Images/Siemens-PLM-Teamcenter-Service-Oriented-Architecture-wp_tcm68-24383.pdf

[Sin14] SINDERMANN, S.: Schnittstellen und Datenaustauschformate. In: Eigner, M.; Roubanov, D.; Zafirov, R.: Modellbasierte Virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 327–347 – ISBN: 978-3-662-43815-2, DOI: 10.1007/978-3-662-45874-7_14

[SK14] SCHWARZER, B.; KRCMAR, H.: Wirtschaftsinformatik – Grundlagen betrieblicher Informationssysteme. Schäffer-Poeschel, 5. Auflage, 2014 – ISBN: 978-3-7910-3397-6

[SL12] SCHWEGMANN, A.; LASKE, M.: Istmodellierung und Istanalyse. In: Becker, J.; Kugeler, M.; Rosemann, M.: Prozessmanagement – Ein Leitfaden zur prozessorientierten Organisationsgestaltung. Springer Gabler, 7. Auflage, 2012, S. 165–194 – ISBN: 978-3-642-33843-4, DOI: 10.1007/978-3-642-33844-1_5

[Sno14] SNOECK, M.: Enterprise Information Systems Engineering – The MERODE Approach. Springer International, 2014 – ISBN: 978-3-319-10144-6, DOI: https://doi.org/10.1007/978-3-319-10145-3

[Sod15] SODEN, M.: Dynamische Modellanalyse von Metamodellen mit operationaler Semantik. Dissertation, 2015, DOI: http://dx.doi.org/10.18452/17168

[SS12] SPECK, M.; SCHNETGÖKE, N.: Sollmodellierung und Prozessoptimierung. In: Becker, J.; Kugeler, M.; Rosemann, M.: Prozessmanagement – Ein Leitfaden zur prozessorientierten Organisationsgestaltung. Springer Gabler, 7. Auflage, 2012, S. 195–228 – ISBN: 978-3-642-33843-4, DOI: 10.1007/978-3-642-33844-1_6

[SS13] SCHMELZER, H. J.; SESSELMANN, W.: Geschäftsprozessmanagement in der Praxis – Kunden zufrieden stellen – Produktivität steigern – Wert erhöhen. Carl Hanser, 8. Auflage, 2013 – ISBN: 978-3-446-43460-8

[SSR12] SCHUH, G.; STICH, V.; RUNGE, S.: Einführung. In: Schuh, G.; Stich, V.: Produktionsplanung und -steuerung 1 – Evolution der PPS. Springer Vieweg, Berlin, 4. Auflage, 2012, S. 3–7 – ISBN: 978-3-642-25422-2, DOI: 10.1007/978-3-642-25423-9_1

[ST07] SAATY, T. L.; TRAN, L. T.: On the invalidity of fuzzifying numerical judgments in the Analytic Hierarchy Process. Mathematical and Computer Modelling 46 (7-8), Elsevier, Oct. 2007, pp. 962–975

[Sta15] STARK, J.: Product Lifecycle Management (Volume 1) – 21st Century Paradigm for Product Realisation. Springer, 3rd Edition, 2015 – ISBN: 978-3-319-17439-6, DOI: https://doi.org/10.1007/978-3-319-17440-2

[Sta22] STARK, R.: Virtual Product Creation in Industry – The Difficult Transformation from IT Enabler Technology to Core Engineering Competence. Springer, Berlin, 2022 – ISBN: 978-3-662-64301-3, DOI: 10.1007/978-3-662-64301-3

[Sta73] STACHOWIAK, H.: Allgemeine Modelltheorie. Springer, 1973 – ISBN: 3-211-81106-0

[STH20a-ol] SCHREPP, M.; THOMASCHEWSKI, J.; HINDERKS, A.: User Experience Questionnaire – Data Analysis Tools, Abgerufen am 24.3.2020, 2020, Unter: https://www.ueq-online.org/Material/Data_Analysis_Tools.zip (besucht am 24. 03. 2020)

[STH20b-ol] SCHREPP, M.; THOMASCHEWSKI, J.; HINDERKS, A.: User Experience Questionnaire – Questionnaire, Abgerufen am 16.5.2020, 2020, Unter: https://www.ueq-online.org/Material/UEQ_All_Languages.zip (besucht am 16. 05. 2020)

[SV11] SCHÜTTE, R.; VERING, O.: Erfolgreiche Geschäftsprozesse durch moderne Warenwirtschaftssysteme – Produktübersicht marktführender Systeme und Auswahlprozess. Springer, 2011 – ISBN: 978-3-642-20522-4, DOI: https://doi.org/10.1007/978-3-642-20523-1

[SVE+07] STAHL, T.; VÖLTER, M.; EFFTINGE, S.; HAASE, A.; Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag, 2. Auflage, 2007

[SW02] SCHABACKER, M.; WOHLBOLD, L.: Benefit Evaluation of EDM/PDM Systems. In: Pawar, K.; Weber, F.; Thoben, K.-D. (Eds.), Proceedings of the 8th International Conference on Concurrent Enterprising. Rome, Italy, 17-19 June 2002, 2002

[SW18] SCHALLMO, D. R. A.; WILLIAMS, C. A.: Digital Transformation Now! – Guiding the Successful Digitalization of Your Business Model. Springer, 2018 – ISBN: 978-3-319-72843-8, DOI: https://doi.org/10.1007/978-3-319-72844-5

[Tab06] TABELING, P.: Softwaresysteme und ihre Modellierung – Grundlagen, Methoden und Techniken. Springer, 2006, DOI: https://doi.org/10.1007/3-540-29276-4

[TG18] TÖRNGREN, M.; GROGAN, P. T.: How to Deal with the Complexity of Future Cyber-Physical Systems? Designs 2 (4), 2018

[Tie13a] TIEMEYER, E.: Enterprise Architecture Management (EAM) – IT-Architekturen erfolgreich planen und steuern. In: Tiemeyer, E.: Handbuch IT-Management – Konzepte, Methoden, Lösungen und Arbeitshilfen für die Praxis. Carl Hanser, 5. Auflage, 2013, S. 89–145 – ISBN: 978-3-446-43557-5, DOI: https://doi.org/10.3139/9783446436220

[Tie13b] TIEMEYER, E.: IT-Management – Herausforderungen und Rollenverständnis heute. In: Tiemeyer, E.: Handbuch IT-Management – Konzepte, Methoden, Lösungen und Arbeitshilfen für die Praxis. Carl Hanser, 5. Auflage, 2013, S. 1–45 – ISBN: 978-3-446-43557-5, DOI: https://doi.org/10.3139/9783446436220

[TMM+22] TRÖLS, M. A.; MARCHEZAN, L.; MASHKOOR, A.; EGYED, A.: Instant and global consistency checking during collaborative engineering. Software and Systems Modeling 21 (6), 2022, S. 2489–2515

[TOG18] THE OPEN GROUP: The Open Group Architecture Framework – Version 9.2, Aug. 2018

[TOG19] THE OPEN GROUP: ArchiMate 3.1 Specification, Nov. 2019

[TW15] THOME, R.; WINKELMANN, A.: Grundzüge der Wirtschaftsinformatik. Springer Gabler, 2015 – ISBN: 978-3-662-46731-2, DOI: 10.1007/978-3-662-46732-9

[UP88] ULRICH, H.; PROBST, G. J.: Anleitung zum ganzheitlichen Denken und Handeln – Ein Brevier für Führungskräfte. Verlag Paul Haupt, 1988

[VAC+09] VOGEL, O.; ARNOLD, I.; CHUGHTAI, A.; IHLER, E.; KEHRER, T.; MEHLIG, U.; ZDUN, U.: Software-Architektur – Grundlagen – Konzepte – Praxis. Spektrum Akademischer Verlag, 2. Auflage, 2009 – ISBN: 978-3-8274-1933-0, DOI: 10.1007/978-3-8274-2267-5

[Var90] VARGAS, L. G.: An overview of the analytic hierarchy process and its applications. European Journal of Operational Research 48 (1), Elsevier, Sept. 1990, pp. 2–8

[Vol14] VOLKMANN, J. W.: Methode zur Konzeption und zur Bestimmung der Wirtschaftlichkeit des Einsatzes von Digital Engineering-Systemen, Fakultät Konstruktions-, Produktions- und Fahrzeugtechnik, Universität Stuttgart, 2014, – ISBN: 978-3-8396-0808-1

[VWZ+18] VAJNA, S.; WEBER, C.; ZEMAN, K.; HEHENBERGER, P.; GERHARD, D.; WARTZACK, S.: CAx für Ingenieure – Eine praxisbezogene Einführung. Springer Vieweg, 3. Auflage, 2018 – ISBN: 978-3-662-54623-9, DOI: 10.1007/978-3-662-54624-6

[WC18] WYNN, D. C.; CLARKSON, P. J.: Process models in design and development. Research in Engineering Design 29 (2), Apr. 2018, pp. 161–202

[Wei14] WEILKIENS, T.: Systems Engineering mit SysML/UML – Modellierung, Analyse, Design. dpunkt.verlag, 3. Auflage, 2014 – ISBN: 978-3-86490-091-4

[Wie98] WIESE, J.: Ein Entscheidungsmodell für die Auswahl von Standardanwendungssoftware am Beispiel von Warenwirtschaftssystemen, Becker, J.; Grob, H. L.; Klein, S., Arbeitsberichte des Instituts für Wirtschaftsinformatik, 1998, Unter: https://miami.uni-muenster.de/Record/e4d2964f-94d8-4c38-b92f-5360049f2981

[WLF14] WINTER, R.; LEGNER, C.; FISCHBACH, K.: Introduction to the special issue on enterprise architecture management. Information Systems and e-Business Management 12 (1), Feb. 1, 2014, pp. 1–4

[WLR+15] WEILKIENS, T.; LAMM, J. G.; ROTH, S.; WALKER, M.: Model-Based System Architecture. John Wiley & Sons, 2015 – ISBN: 978-1-118-89364-7, DOI: 10.1002/9781119051930

[WRF+15] WALDEN, D. D.; ROEDLER, G. J.; FORSBERG, K. J.; HAMELIN, R. D.; SHORTELL, T. M. (Eds.): INCOSE Systems Engineering Handbook – A Guide for System Life Cycle Processes and Activities. Wiley, 4th Edition, 2015 – ISBN: 978-1-118-99940-0

[WS18] WOLF, T.; STROHSCHEN, J.-H.: Digitalisierung – Definition und Reife. Informatik Spektrum 41 (1), 2018, S. 56–64

[Zaf14] ZAFIROV, R.: Modellbildung und Spezifikation. In: Eigner, M.; Roubanov, D.; Zafirov, R.: Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, 2014, S. 77–96 – ISBN: 978-3-662-43816-9, DOI: 10.1007/978-3-662-43816-9_4

[Zan14] ZANGEMEISTER, C.: Nutzwertanalyse in der Systemtechnik – Eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen. Zangemeister & Partner, 5. Auflage, 2014 – ISBN: 978-3-923264-00-1

[Zic13] ZICKERT, G.: Elektrokonstruktion – Gestaltung, Schaltpläne und Engineering mit EPLAN. Carl Hanser, 3. Auflage, 2013

[Zwi67] ZWICKY, F.: The Morphological Approach to Discovery, Invention, Research and Construction. In: Zwicky, F.; Wilson, A. G.: New Methods of Thought and Procedure. Springer, 1967, pp. 273–29 – ISBN: 978-3-642-87619-6, DOI: 10.1007/978-3-642-87617-2_14

Normen und Richtlinien

[DIN60050] DIN IEC 60050-351 – Internationales Elektrotechnisches Wörterbuch – Teil 351: Leittechnik, Sep. 2014

[DIN61508] DIN EN 61508-3; VDE 0803-3 – Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme – Teil 3: Anforderungen an Software, Feb. 2011

[DIN9241-11] DIN EN ISO 9241-11 – Ergonomie der Mensch-System-Interaktion – Teil 11: Gebrauchstauglichkeit: Begriffe und Konzepte, Nov. 2018

[ISO10303] ISO 10303-242 – Industrial automation systems and integration – Product data representation and exchange – Part 242: Application protocol: Managed model-based 3D engineering, Dec. 2014

[ISO15288] ISO/IEC/IEEE 15288 – Systems and software engineering – System life cycle processes, May 2015

[ISO19505-1] ISO/IEC 19505-1 – Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 1: Infrastructure, Apr. 2012

[ISO19505-2] ISO/IEC 19505-2 – Information technology – Object Management Group Unified Modeling Language (OMG UML) – Part 2: Superstructure, Apr. 2012

[ISO19510] ISO/IEC 19510 – Information technology – Object Management Group Business Process Model and Notation, July 2013

[ISO19514] ISO/IEC 19514 – Information technology – Object management group systems modeling language (OMG SysML), Mar. 2017

[ISO19759] ISO/IEC TR 19759 – Software Engineering – Guide to the software engineering body of knowledge (SWEBOK), Oct. 2015

[ISO21778] ISO/IEC 21778 – Information technology – The JSON data interchange syntax, Nov. 2017

[ISO24765] ISO/IEC/IEEE 24765 – Systems and software engineering – Vocabulary, Piscataway, NJ, USA, Aug. 2017

[ISO25010] ISO/IEC 25010 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, Mar. 2011

[ISO29148] ISO/IEC/IEEE 29148 – Systems and software engineering – Life cycle processes – Requirements engineering, Nov. 2018

[ISO42010] ISO/IEC/IEEE 42010 – Systems and software engineering – Architecture description, Dec. 2011

[ISO9000] DIN EN ISO 9000 – Qualitätsmanagementsysteme – Grundlagen und Begriffe, Berlin, 2015-11

[VDI2206] VDI 2206 – Entwicklungsmethodik für mechatronische Systeme, Juni 2015

[VDI2219] VDI 2219 – Informationsverarbeitung in der Produktentwicklung – Einführung und Betrieb von PDM-Systemen, Sep. 2016

[VDI3695-4] VDI/VDE 3695-4 – Engineering von Anlagen – Evaluieren und optimieren des Engineerings – Themenfeld Hilfsmittel, Dez. 2010

Anhang

Inhaltsverzeichnis

Seite

A1 Ergänzungen zur Spezifikationstechnik A-1

A1.1 Festlegungen zur grafischen Notation A-1

A1.2 Zuordnung der Partialmodelle zu Viewpoints A-2

A1.3 Profil der Spezifikationstechnik A-3

A2 Beschreibung der Viewpoints A-5

A2.1 Viewpoints der Prozess Domäne A-5

A2.2 Viewpoints der Methoden Domäne A-6

A2.3 Viewpoints der Funktions Domäne A-7

A2.4 Viewpoints der Applikations Domäne A-8

A2.5 Viewpoints der Anforderungs Domäne A-10

A3 Methodische Hilfsmittel A-13

A3.1 Paarweiser Vergleich mittels AHP A-13

A3.2 Bewertung der User Experience anhand des UEQ A-15

A4 Übersetzungen aus dem Englischen A-18

A1 Ergänzungen zur Spezifikationstechnik

A1.1 Festlegungen zur grafischen Notation

Die Festlegungen zur grafischen Notation sollen eine reproduzierbare Nutzung der Modellierungssprache ermöglichen. Nachfolgend werden die Farben zur Notation von Modellelementen, gegliedert nach den Domänen bzw. Problemstellungen, definiert.

Tabelle A-1: Farbliche Darstellung der Sprachkonstrukte

Domäne	Zugeordnete Sprachelemente	Farbcode
Übergreifende Dom.	z.B. Artefakte, Attribute, Methoden, Technische Ressourcen, Rollen[1], Personen, Potentiale, Dokumentationsobjekte	000R 000G 000B
Prozess Dom.	z.B. Aktivitäten, Lebenszyklusmanagement-Prozesse,	255R 231G 173B
Methoden Dom.	z.B. Aufgaben, Methodenschritte,	255R 231G 173B
Funktions Dom.	z.B. Funktionsgruppe, Funktion	224R 218G 235B
IT-Applikations Dom.	z.B. Softwaregruppe, Softwarekomponente	206R 232G 247B
Anforderungs Dom.	z.B. Anforderungen, Anwendungsfälle[2]	252R 229G 231B

[1] Rollen werden zur Differenzierung in 208R 209G 211B dargestellt [2] Anwendungsfälle werden in 255R 231G 173B dargestellt

Die konkrete Syntax der einzelnen Sprachkonstrukte ist in Kap. 5.3.2 definiert. Entsprechende Symbole, z.B. zur Implementierung in Modellierungswerkzeuge, die nicht durch die UML, SysML, BPMN oder SPEM bereitgestellt oder vererbt werden, befinden sich auf dem der Arbeit beiliegenden Datenträger.

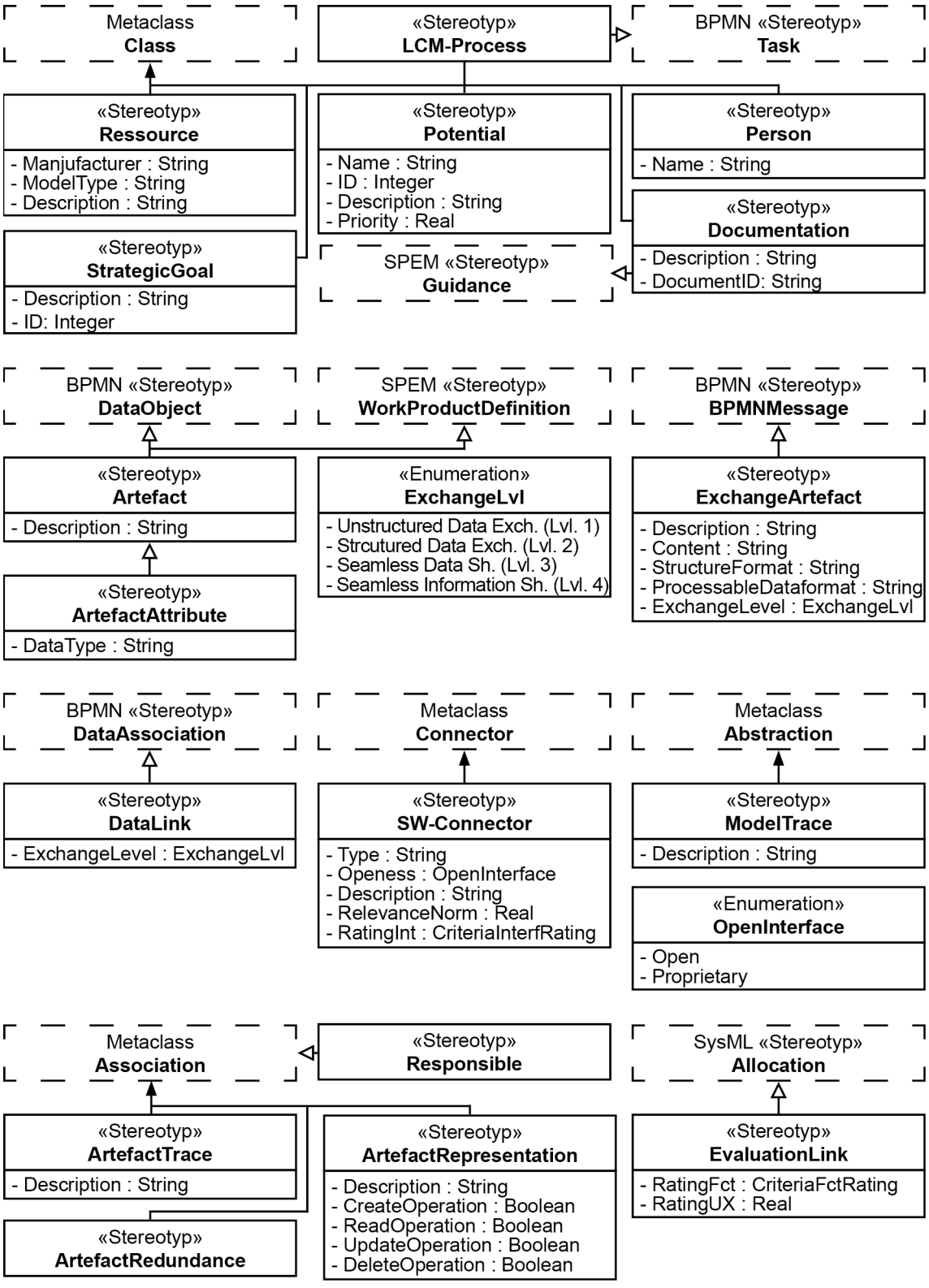
A1.2 Zuordnung der Partialmodelle zu Viewpoints

Tabelle A-2: Schnittmengen zwischen Partialmodellen und Viewpoints

			Partialmodell						
Viewpoint ID	**Viewpoint Bezeichnung**	**Partialmodell**	Prozessmodell	Methodenmodell	Use Cases	Anforderungen	Funktionale Arch.	Lösungsmodell	Kompon. Arch.
PPV01	Organisations Umfeld		X						
PPV02	Prozessablauf		X						
PPV03a	Artefaktliste		X						
PPV03b	Ressourcenliste		X	(X)					
PPV04a	Prozessschnittstellen		X						
PPV04b	Prozesspotentiale		X						
PPV05	Lebenszyklusmgt.-Mpg		X						
PMV02	Methodendiagramm			X					
PMV03	Artefaktdekomposition		(X)	X					
PMV04a	Aktivitäts-Methoden-Mpg.		X						
PMV04b	Artefaktbeziehungen			X					
PMV04c	Methodenpotentiale			X					
PFV01	Funktions Umfeld						X		
PFV03	Funktionale Dekomposition						X		
PFV04a	Funktionale Architektur						X		
PFV04b	Funktionale Schnittstellen						X		
PFV05	Funktions-Traceability-Mpg.						(X)		
PAV01	Applikations Umfeld								X
PAV03a	Architektur Dekomposition								X
PAV03b	SW-Komponenten-Zuordnung							X	
PAV04a	Applikationsarchitektur								X
PAV04b	Software Schnittstellen								X
PAV05a	SW-Funktions-Mpg.							X	
PAV05b	Variantendefinition & -mgt.							X	
PRV01a	Rollenidentifikation		X	X	X	(X)			
PRV01b	Personenidentifikation		(X)	(X)	(X)	(X)			
PRV01b	Strategische Ziele		X	X	X	X	X	X	X
PRV02	Anwendungsfälle				X				
PRV03a	Anf.-Funktionszuordnung					X			
PRV03b	Anwendungsfall-Funktionszuordnung				X				
PRV03c	Potentialliste		X	X	X	X	X	X	X
PRV05a	Anf.-Traceability-Mpg.					(X)			
PRV05b	Lösungsbewertung								X
PRV05c	Rollen-Personen-Mpg.					(X)			

Legende X: Vorgesehen (X): Möglich Mpg.: Mapping Mgt.: Management

A1.3 Profil der Spezifikationstechnik



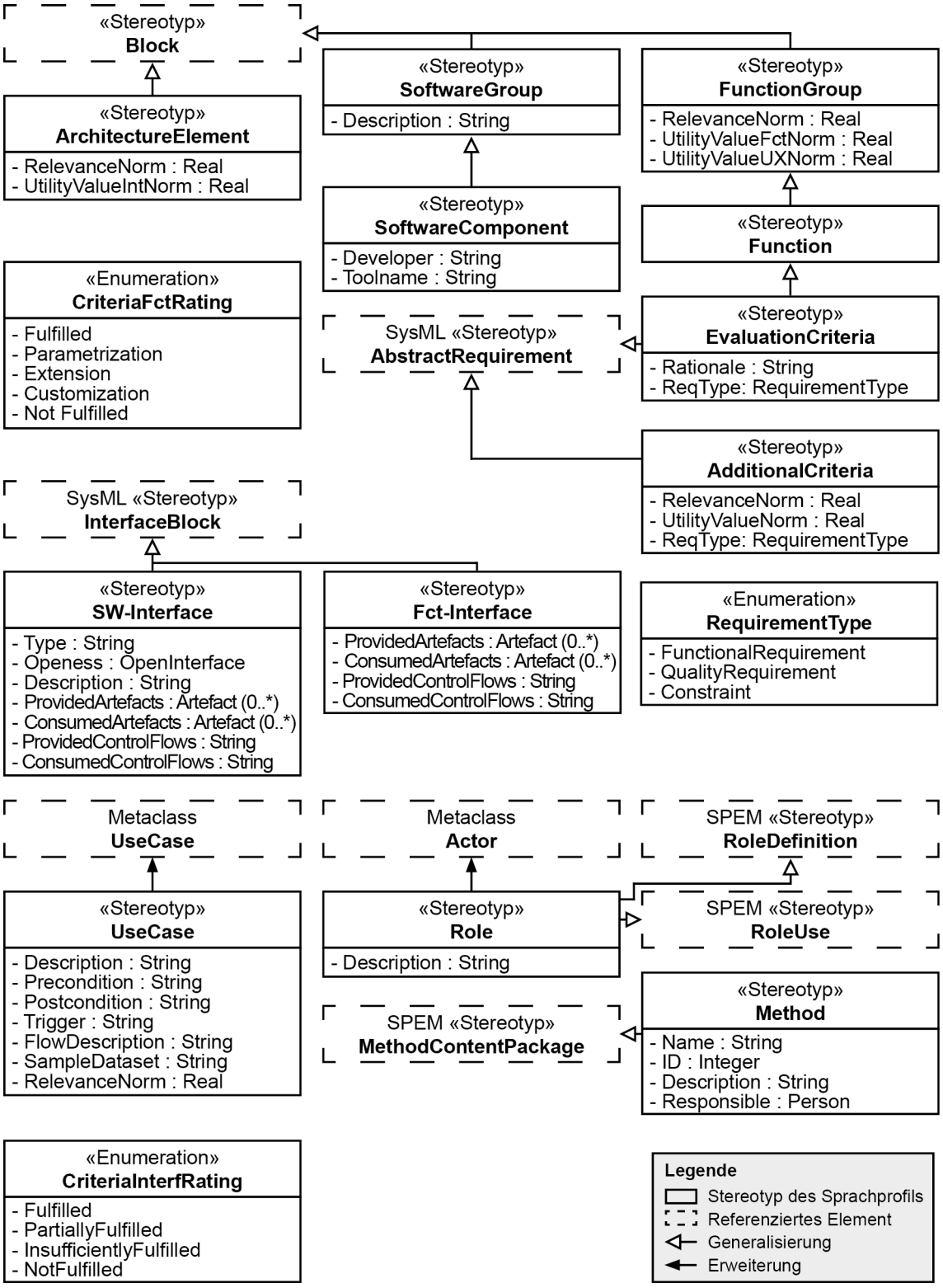


Bild A-1: Bestandteile und Gliederung der Spezifikationstechnik im Überblick

A2 Beschreibung der Viewpoints

Optionale oder selten genutzte Elemente werden in *kursiver Schrift* dargestellt.

A2.1 Viewpoints der Prozess Domäne

PPV01 – Organisations Umfeld

Kurzbeschreibung: Beschreibt Umfeldelemente, die über Artefaktaustauschflüsse eine Relation zu dem Engineering-System aufweisen.

Enthaltene Sprachkonstrukte: Umfeldelement, Engineering-System, Austauschartefakt, Artefaktaustauschfluss

Typische Darstellungsform: Prozessdiagramm

PPV02 – Prozessablauf

Kurzbeschreibung: Beschreibt die sachlogischen Zusammenhänge eines Prozesses.

Enthaltene Sprachkonstrukte: Swimlane, Aktivität, Artefakt, *Technische Ressource*, Sequenzfluss, Datenassoziation, Assoziation

Typische Darstellungsform: Prozessdiagramm

PPV03a – Artefaktliste

Kurzbeschreibung: Listet alle Artefakte aus der Prozessebene auf.

Enthaltene Sprachkonstrukte: Artefakte, Austauschartefakt

Typische Darstellungsform: Tabelle (Spalten: (1) Artefaktname; (2) Artefaktbeschreibung; (3) Auflistung der Modellelemente vom Typ Aktivität, die das Artefakt als Output nutzen; (4) Auflistung der Modellelemente vom Typ Aktivität, die das Artefakt als Input nutzen (5) Artefakttyp)

PPV03b – Ressourcenliste

Kurzbeschreibung: Listet alle Ressourcen aus der Prozessebene auf.

Enthaltene Sprachkonstrukte: Ressourcen, Aktivitäten

Typische Darstellungsform: Tabelle (Spalten: (1) Ressourcenname; (2) Ressourcenbeschreibung; (3) Auflistung der Modellelemente vom Typ Aktivität, welche die Ressource verwenden)

PPV04a – Prozessschnittstellen

Kurzbeschreibung: Zeigt alle Schnittstellen auf, bei denen eine Aktivität ein Artefakt als Input nutzt.

Enthaltene Sprachkonstrukte: Datenassoziation, Artefakt, Austauschartefakt, Aktivität

Typische Darstellungsform: Tabelle (Spalten: (1) Datenassoziationsname; (2) Artefakt (Inputobjekt); (3) Aufnehmende Aktivität; (4) Auflistung der Aktivitäten, die das Artefakt erzeugen; (5) Automatisierungsgrad des Datenaustauschs))

PPV04b – Prozesspotentiale

Kurzbeschreibung: Beschreibt alle Potentiale, die sich auf Modellelemente der Prozessebene beziehen.

Enthaltene Sprachkonstrukte: Potentiale, verknüpfte Elemente

Typische Darstellungsform: Tabelle (Spalten: (1) Potentialname; (2) Potentialbeschreibung; (3) Auflistung der Modellelemente, auf die sich das Potential bezieht)

PPV05 – Lebenszyklusmanagement-Mapping

Kurzbeschreibung: Ordnet die durch ihn gesteuerten Artefakte einem Lebenszyklusmanagement-Prozess zu.

Enthaltene Sprachkonstrukte: Artefakt, Lebenszyklusmanagement-Prozess, Assoziation

Typische Darstellungsform: Matrix (Zeilen: Artefakte / Spalten: Lebenszyklusmanagement-Prozess / Zelle: Assoziation)

A2.2 Viewpoints der Methoden Domäne

PMV02 – Methodendiagramm

Kurzbeschreibung: Beschreibt die Tätigkeiten und resultierenden Artefakte einer Methode.

Enthaltene Sprachkonstrukte: Aufgabe, Methodenschritt, Artefakt, Input-/Outputbeziehung

Typische Darstellungsform: Methodendiagramm

PMV03 – Artefaktdekomposition

Kurzbeschreibung: Beschreibt die hierarchische Struktur von Artefakten.

Enthaltene Sprachkonstrukte: Artefakte, Attribute, Aggregation

Typische Darstellungsform: Methodendiagramm

PMV04a – Aktivitäts-Methoden-Mapping

Kurzbeschreibung: Ordnet eine Methode einer Aktivität auf Prozessebene zu, bei deren Ausführung die Methode genutzt wird bzw. werden kann.

Enthaltene Sprachkonstrukte: Aktivität, Methode

Typische Darstellungsform: Matrix (Zeilen: Aktivität / Spalten: Methode / Zelle: Assoziation)

PMV04b – Artefaktbeziehungen

Kurzbeschreibung: Beschreibt die aufgrund methodischer Vorgaben oder Redundanz bzw. Instanzierung resultierenden Beziehungen zwischen Artefakten.

Enthaltene Sprachkonstrukte: Artefakt, Attribut, Aggregation, Artefaktabhängigkeit, Artefaktredundanz, Artefaktrepräsentation

Typische Darstellungsform: Matrix (Zeilen: Artefakte, Attribute / Spalten: Artefakte, Attribute / Zelle: Artefaktabhängigkeit, Artefaktredundanz, Artefaktrepräsentation)

PMV04c – Methodenpotentiale

Kurzbeschreibung: Beschreibt alle Potentiale, die sich auf Modellelemente der Methodenebene beziehen.

Enthaltene Sprachkonstrukte: Potentiale, verknüpfte Elemente

Typische Darstellungsform: Tabelle (Spalten: (1) Potentialname; (2) Potentialbeschreibung; (3) Auflistung der Modellelemente, auf die sich das Potential bezieht)

A2.3 Viewpoints der Funktions Domäne

PFV01 – Funktions Umfeld

Kurzbeschreibung: Beschreibt den Kontext der als Black-Box betrachteten Hauptfunktion einer betrachteten funktionalen Architektur.

Enthaltene Sprachkonstrukte: Funktionen (Notwendige Funktionen im Kontext), Funktionen (Hauptfunktion der funktionalen Architektur als Black-Box), Beziehung, Artefaktfluss, Port

Typische Darstellungsform: Internes Blockdiagramm

PFV03 – Funktionale Dekomposition

Kurzbeschreibung: Beschreibt die hierarchische Struktur von Funktionen.

Enthaltene Sprachkonstrukte: Funktion, Aggregation

Typische Darstellungsform: Blockdefinitionsdiagramm

PFV04a – Funktionale Architektur

Kurzbeschreibung: Beschreibt die Relationen zwischen Funktionen des Zielsystems.

Enthaltene Sprachkonstrukte: Funktionen, Beziehung, Artefaktfluss, Port

Typische Darstellungsform: Internes Blockdiagramm

PFV04b – Funktionale Schnittstellen

Kurzbeschreibung: Listet alle Funktions-Schnittstellen auf und charakterisiert diese.

Enthaltene Sprachkonstrukte: Funktions-Schnittstellen

Typische Darstellungsform: Blockdefinitionsdiagramm

PFV05 – Funktions-Traceability-Mapping

Kurzbeschreibung: Stellt die Rückverfolgbarkeit der Funktionen zu den sie bedingenden Modellelementen her.

Enthaltene Sprachkonstrukte: Funktion, beliebiges Modellelement

Typische Darstellungsform: Matrix (Zeilen: Alle Modellelemente des betrachteten Modellumfangs / Spalten: Funktionen / Zelle: Model-Trace)

A2.4 Viewpoints der Applikations Domäne

PAV01 – Applikations Umfeld

Kurzbeschreibung: Beschreibt die IT-Applikationen im Umfeld des zu betrachtenden Systems und ihre Beziehungen zur als Black-Box betrachteten Applikationsarchitektur.

Enthaltene Sprachkonstrukte: Softwarekomponente (Kontextelemente), Architekturelement (Black-Box der Applikationsarchitektur des betrachteten Systems), Beziehung, Artefaktfluss, Port

Typische Darstellungsform: Internes Blockdiagramm

PAV03a – Architektur Dekomposition

Kurzbeschreibung: Beschreibt den hierarchischen Aufbau der Applikationsarchitektur einer Lösungsvariante.

Enthaltene Sprachkonstrukte: Softwarekomponente, Architekturelement, Aggregation

Typische Darstellungsform: Blockdefinitionsdiagramm

PAV03b – Software-Komponenten-Zuordnung

Kurzbeschreibung: Beschreibt die Zuordnung von Softwarekomponenten zu Softwaregruppen.

Enthaltene Sprachkonstrukte: Softwaregruppe, Softwarekomponente

Typische Darstellungsform: Matrix (Zeilen: Softwarekomponente / Spalten: Softwaregruppe / Zelle: Assoziation)

PAV04a – Applikationsarchitektur

Kurzbeschreibung: Beschreibt die Relationen zwischen Softwarekomponenten und Architekturelementen des Zielsystems.

Enthaltene Sprachkonstrukte: Softwarekomponente, Architekturelement, Schnittstellenfluss, Artefaktfluss, Schnittstellenport

Typische Darstellungsform: Internes Blockdiagramm

PAV04b – Software Schnittstellen

Kurzbeschreibung: Listet alle Software-Schnittstellen auf und charakterisiert diese.

Enthaltene Sprachkonstrukte: Software-Schnittstellen

Typische Darstellungsform: Blockdefinitionsdiagramm

PAV05a – Software-Funktions-Mapping

Kurzbeschreibung: Beschreibt die Umsetzung von Funktionen durch Softwarekomponenten als Allokationssicht.

Enthaltene Sprachkonstrukte: Softwarekomponenten, Funktionen

Typische Darstellungsform: Matrix (Zeilen: Softwarekomponenten / Spalten: Funktionen / Zelle: Evaluierungsbeziehung)

PAV05b – Variantendefinition & -management

Kurzbeschreibung: Beschreibt die Definition verschiedener Lösungsvarianten.

Enthaltene Sprachkonstrukte: Lösungsvarianten[1]

Typische Darstellungsform: Klassendiagramm

A2.5 Viewpoints der Anforderungs Domäne

PRV01a – Rollenidentifikation

Kurzbeschreibung: Beschreibt die im Modell enthaltenen Rollen.

Enthaltene Sprachkonstrukte: Rolle

Typische Darstellungsform: Tabelle (Spalten: (1) Name der Rolle; (2) Rollenbeschreibung)

PRV01b – Personenidentifikation

Kurzbeschreibung: Beschreibt die im Modell enthaltenen Personen.

Enthaltene Sprachkonstrukte: Rolle, Person

Typische Darstellungsform: Tabelle (Spalten: (1) Name der Person; (2) Kommentar; (3) Mit der Person verknüpfte Rollen)

PRV01c – Strategische Ziele

Kurzbeschreibung: Listet alle strategischen Ziele auf.

Enthaltene Sprachkonstrukte: Strategische Ziele, beliebiges Modellelement

Typische Darstellungsform: Tabelle (Spalten: (1) Name des strategischen Ziels; (2) Beschreibung des strategischen Ziels; (3) Auflistung der Modellelemente, die mit dem strategischen Ziel verknüpft sind)

PRV02 – Anwendungsfälle

Kurzbeschreibung: Beschreibt die Anwendungsfälle und ihre Zuordnung zu einer Rolle.

Enthaltene Sprachkonstrukte: Anwendungsfall, Rolle, Assoziation (ungerichtet)

Typische Darstellungsform: Use Case Diagramm

---

[1] Die notwendigen Sprachelemente werden typischerweise über das Modellierungswerkzeug bereitgestellt, da diese das Variantenmanagement steuern.

PRV03a – Anforderungs-Funktions-Zuordnung

Kurzbeschreibung: Beschreibt die Einordnung der Anforderungen in die funktionale Dekomposition zum Aufbau des Zielsystems für die funktionale Eignung.

Enthaltene Sprachkonstrukte: Funktionsgruppe, Funktion, Anwendungsfall, Aggregation, Enthält-Beziehung

Typische Darstellungsform: Blockdefinitionsdiagramm

PRV03b – Anwendungsfall-Funktions-Zuordnung

Kurzbeschreibung: Beschreibt die Einordnung der Anwendungsfälle in die funktionale Dekomposition zum Aufbau des Zielsystems für die Gebrauchstauglichkeit.

Enthaltene Sprachkonstrukte: Funktionsgruppe, Funktion, Anwendungsfall, Aggregation, Enthält-Beziehung

Typische Darstellungsform: Blockdefinitionsdiagramm

PRV03c – Potentialliste

Kurzbeschreibung: Listet alle Potentiale auf.

Enthaltene Sprachkonstrukte: Potential, beliebiges Modellelement

Typische Darstellungsform: Tabelle (Spalten: (1) Name des Potentials; (2) Beschreibung des Potentials; (3) Auflistung der Modellelemente, die mit dem Potential verknüpft sind)

PRV05a – Anforderungs-Traceability-Mapping

Kurzbeschreibung: Stellt die Rückverfolgbarkeit der Anforderungen/UseCases zu den sie bedingenden Modellelementen her.

Enthaltene Sprachkonstrukte: Anforderungen, Anwendungsfall, beliebiges Modellelement

Typische Darstellungsform: Matrix (Zeilen: Alle Modellelemente des betrachteten Modellumfangs / Spalten: Anforderungen oder Anwendungsfall / Zelle: Model-Trace)

PRV05b – Lösungsbewertung

Funktionale Eignung bzw. UX

Kurzbeschreibung: Beschreibt die Bewertung von Kriterien in Bezug auf die Softwarekomponenten einer Lösungsvariante.

Enthaltene Sprachkonstrukte: Softwarekomponente, Architekturelement, Funktionsgruppe, Funktion, Bewertungskriterium, Anwendungsfall

Typische Darstellungsform: Tabelle (Spalten: (1) Hierarchie der Funktionsgruppen bzw. Funktionen; (2) Modellelement der umsetzenden Softwarekomponente; (3) Normierte Relevanz *RelevanceNorm* als Attributwert der Funktionsgruppe, der Funktion bzw. des Bewertungskriteriums/Anwendungsfalls; (4) Relative Relevanz der Funktionsgruppe, der Funktion bzw. des Bewertungskriteriums/Anwendungsfalls; (5) Evaluierungsergebnis als Attributwert *RatingFct* bzw. *RatingUX* der Evaluierungsbeziehung; (6) Normierter Teilnutzenwert als Attribut *UtilityValueFctNorm* bzw. *UtilityValueUXNorm* als Attributwert der Funktionsgruppe, der Funktion bzw. des Bewertungskriteriums/Anwendungsfalls) (7) Relativer Teilnutzenwert der Funktionsgruppe, der Funktion bzw. des Bewertungskriteriums/Anwendungsfalls

Interoperabilität

Kurzbeschreibung: Beschreibt die Bewertung von Kriterien in Bezug auf die Softwarekomponenten einer Lösungsvariante.

Enthaltene Sprachkonstrukte: Architekturelement, Schnittstellenfluss

Typische Darstellungsform: Tabelle (Spalten: (1) Architekturelemente und Schnittstellenfluss einer Ebene; (2) Normierte Relevanz *RelevanceNorm* als Attributwert des Architekturelements bzw. des Schnittstellenflusses; (3) Relative Relevanz des Architekturelements bzw. Schnittstellenflusses; (4) Evaluierungsergebnis als Attributwert *RatingInt* des Schnittstellenflusses; (6) Normierter Teilnutzenwert als Attribut *UtilityValueIntNorm* als Attributwert des Teilarchitekturelements; (7) Relativer Teilnutzenwert des Teilarchitekturelements

PRV05c – Rollen-Personen-Mapping

Kurzbeschreibung: Zuordnung von Rollen zu den Personen, die diese Rolle ausführen.

Enthaltene Sprachkonstrukte: Rolle, Person

Typische Darstellungsform: Matrix (Zeilen: Rolle / Spalten: Person / Zelle: Assoziation)

A3 Methodische Hilfsmittel

Nachfolgend sollen methodische Hilfsmittel detaillierter vorgestellt werden, um sie im Kontext der Spezifikationstechnik zu nutzen. Die Einsatzbereiche der Hilfsmittel sind in Kap. 5.3.2 erläutert. In Anhang A3.1 wird der paarweise Vergleich mittels des Analytic Hierarchy Process zur Priorisierung von Items erläutert. In Anhang A3.2 wird die Bewertung der User Experience anhand des User Experience Questionaire Fragebogens vorgestellt.

A3.1 Paarweiser Vergleich mittels AHP

Im Rahmen der Spezifikationstechnik müssen an verschiedenen Stellen Elemente priorisiert werden, um einen Gewichtungsfaktor zu erhalten. Dazu eignen sich Teilschritte des Analyitical Hierarchical Process (AHP). Die für die Priorisierung relevanten Schritte sollen im Folgenden kurz erläutert werden. Im Kern sind zwei Phasen der AHP zu unterscheiden. In der *Hierarchiebildung* wird ein Entscheidungsproblem hierarchisch gegliedert. Im Rahmen der Spezifikationstechnik erfolgt dies im Bereich der Partialmodelle. In der *Evaluierung* werden paarweise Vergleiche anhand von Expertenurteilen für die Elemente einer Hierarchieebene durchgeführt, um den Gewichtungsfaktor jedes Elementes zu definieren [Var90, S. 2]. Die AHP erlaubt gegenüber dem einfachen paarweisen Vergleich die Prüfung der Konsistenz der Ergebnisse [Saa08, S. 83].

Zunächst erfolgt die **Ermittlung der Prioritäten** für alle Elemente einer Hierarchiestufe mittels eines paarweisen Vergleichs. Die Elemente werden in eine Tabelle in Form einer Matrix aufgetragen, bei der in den Zeilen i und Spalten k jeweils die zu bewertenden Elemente aufgetragen werden. Für jede Kombination einer Zeile i und Spalte k wird die relative Wichtigkeit von i gegenüber k als ein Vielfaches einer Verhältnisskala bewertet. Diese ist in Tabelle A-3 dargestellt. Die Bewertung der Kombination k zu i muss sich reziprok verhalten, sodass der Vergleichswert hier dem Kehrwert entspricht. Als Ergebnis der Bewertung ergeben sich die ersten vier Zeilen der in Tabelle A-4-A dargestellten Matrix.

Anschließend folgen die Berechnung der Gewichtungsvektoren sowie die Konsistenzbewertung. Die mathematischen Verfahren sind in der Originalfassung in [Saa77, S. 234ff] oder in komprimierter Form bspw. in [Göt14, S. 200ff] umfassend erläutert und werden hier nur verkürzt beschrieben. Zunächst erfolgt die **Berechnung der Gewichtungsvektoren,** um den jeweiligen Gewichtungsfaktor der Elemente zu bestimmen. Es existieren verschiedene mathematische Lösungsansätze, am gebräuchlichsten ist die Berechung des Gewichtungsvektors über den Haupt-Eigenvektor (vgl. [Bru15, S. 17ff]). Hier soll nur auf ein näherungsweises Verfahren eingegangen werden. Dazu werden die Spaltensummen der Bewertungsmatrix gebildet und die Paarvergleichswerte durch die Spaltensumme dividiert, um eine Normierung auf 1 zu erreichen. Die Zeilenelemente der resultierenden normalisier-

ten Matrix werden zu Zeilensummen addiert, die durch die Anzahl der Elemente dividiert werden. Dieser Wert stellt näherungsweise den Gewichtungsfaktor w_k des Elements dar. Das Resultat ist in Tabelle A-4-B dargestellt.

Tabelle A-3: Skala zur Bewertung der Elemente nach [Saa77, S. 246], [Saa13, S. 54]

Wert	Definition	Interpretation
1	**Gleiche Bedeutung (Indifferenz)**	Die verglichenen Elemente i und k haben die gleiche Bedeutung (in Bezug auf das hierarchisch übergeordnete Element).
3	**Etwas größere Bedeutung**	Die Erfahrung und Einschätzung der Experten zeigen eine geringfügig höhere Bedeutung des einen gegenüber dem anderen Element auf.
5	**Erheblich größere Bedeutung**	Die Erfahrung und Einschätzung der Experten zeigen eine deutlich höhere Bedeutung des einen gegenüber dem anderen Element auf.
7	**Sehr viel größere Bedeutung**	Die Erfahrung und Einschätzung der Experten zeigen eine sehr viel höhere Bedeutung auf oder die Dominanz wird in der Praxis nachgewiesen.
9	**Absolut dominierend**	Der größtmögliche Bedeutungsunterschied, der zwischen zwei Elementen auftreten kann.

Zur feineren Differenzierung können Zwischenwerte genutzt werden (z.B. 2, 4, 6, 8 oder 1.1 bis 1.9)

Tabelle A-4: Beispielhaft im paarweisen Vergleich ausgefüllte (A) und normierte (B) Matrizen

A) Bewertungsergebnisse

Element	A	B	C	D
A	1	1/5	1/5	1/7
B	5	1	1	1/5
C	5	1	1	1/6
D	7	6	5	1
Summe	18	8,2	7,2	1,51

B) Normierte Werte & Zeilendurchschnitt

Element	A	B	C	D	w_k
A	0,056	0,024	0,028	0,095	**0,051**
B	0,278	0,122	0,139	0,132	**0,168**
C	0,278	0,122	0,139	0,110	**0,162**
D	0,389	0,732	0,694	0,662	**0,619**

Da der paarweise Vergleich auf Bewertungen von Experten beruht, können sich inkonsistente Bewertungen ergeben. Daher erfolgt eine **Konsistenzbewertung**. Auch hier existieren präzise mathematische Verfahren, im Folgenden wird jedoch ein näherungsweises Verfahren vorgestellt. In Tabelle A-5 wurden in Zeile 2 für die Spalten 2 bis 5 die jeweiligen Gewichtungsfaktoren übernommen. In den Zeilen 3 bis 6 wurden für die Spalten 2 bis 5 von Tabelle A-5 die Bewertungsergebnisse aus Tabelle A-4-A übernommen und mit den Gewichtungsfaktoren in Zeile 2 der jeweiligen Spalte multipliziert. Die Zeilensummen S_z werden für die Zeilen 3 bis 6 berechnet und in die Spalte 6 eingetragen. Anschließend dividiert man für die Zeilen 3 bis 6 jeweils die Zeilensumme S_z durch den Gewichtungsfaktor g, das Ergebnis wird in Spalte 8 eingetragen. Die Summe von Spalte 8 dividiert durch die Anzahl der Elemente n ergibt den maximalen Eigenwert λ_{max}. Aus diesem kann anhand von Formel 3-1 der Konsistenzindex C_I berechnet werden. Um festzustellen, ob die Inkonsistenz noch tolerabel ist, wird C_I mit einem in Tabelle A-6 aufgeführten empirisch ermittelten, durchschnittlichen Konsistenzindex R_I dividiert. Das Resultat ist die Konsistenzverhältniszahl C_R [MP18, S. 13ff], [MK13, S. 123f]. Ein Wert von $C_R \leq 0,1$ gilt

als akzeptabel. Bei größeren Werten sollten die Bewertungen überprüft und überarbeitet werden [Saa13, S. 58].

Tabelle A-5: Berechnung der Konsistenz einer Matrix

Element/Größe	A	B	C	D	sz	w_k	sz/ g
w_k	*0,051*	*0,168*	*0,162*	*0,619*			
A	0,003	0,001	0,001	0,005	**0,010**	0,051	**4,054**
B	0,047	0,020	0,023	0,022	**0,113**	0,168	**4,213**
C	0,045	0,020	0,023	0,018	**0,123**	0,162	**4,229**
D	0,241	0,453	0,430	0,410	**1,535**	0,619	**4,507**
						λ_{max}	**4,251**

$$C_I = \frac{\lambda_max - n}{n-1}$$

Gleichung A-1: Berechnung des Konsistenzindex C_I

Tabelle A-6: Wertetabelle für den durchschnittlichen Konsistenzindex R_I in Abhängigkeit der Matrixgröße n nach [ST07, S. 966]

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R_I	0	0	0,52	0,89	1,11	1,25	1,35	1,40	1,45	1,49	1,52	1,54	1,56	1,58	1,59

Falls die Gewichtung als konsistent gilt, können die Gewichtungsfaktoren w_k als Attribut der gewichteten Modellelemente in das Modell übernommen werden. Für die genauere Berechnung stehen rechnerbasierte Werkzeuge bereit, z.B. das in [Goe18] vorgestellte und unter [Goe19] erreichbare webbasierte Werkzeug AHP-AS.

A3.2 Bewertung der User Experience anhand des UEQ

Der UEQ bewertet die in Tabelle A-7 aufgelisteten 26 bipolaren Items. Jedes Item spannt eine siebenstufige Skala auf, von -3 (stärkste negative Ausprägung) bis +3 (stärkste positive Ausprägung). Eine neutrale Antwort wird durch den Wert 0 widergespiegelt. Jedes Item kann einer der folgenden sechs Skalen zugeordnet werden: Effektivität, Durchschaubarkeit, Vorhersagbarkeit, Stimulation, Originalität und Attraktivität. Die Skalen sind für die Probanden nicht sichtbar. Zum Ausfüllen vorbereitete Fragebögen stehen in verschiedenen Sprachen unter [STH20b-ol] zum Download bereit. Die Bewertung durch die Probanden sollte unmittelbar nach dem Durchlaufen der Anwendungsfälle ausgefüllt werden, um eine gegenseitige Beeinflussung der Probanden auszuschließen. Für das Ausfüllen geben die Autoren folgende Hinweise ab [STH20b-ol], [Sch19, S. 2, 8].

- Durch **Ankreuzen** eines Kreises erfolgt die Bewertung eines Items für das betrachtete Produkt.
- Für jedes Item muss **immer genau eine Bewertung** abgegeben werden. Dies gilt auch, wenn der Proband den Eindruck hat, dass das Gegensatzpaar nicht gut zum bewerteten Produkt passt oder sich unsicher ist.
- Enscheidungen sollen möglichst **spontan** erfolgen, Begriffe sollen nicht lange hinterfragt werden.

Tabelle A-7: Fragebogen des UEQ zum Ausfüllen durch Probanden nach [LHS08, S. 75] mit der für die Probanden nicht sichtbare Zuordnung zu den Skalen

Item	Bewertungspol 1	1	2	3	4	5	6	7	Bewertungspol 2	Skala
1	unerfreulich	○	○	○	○	○	○	○	nicht erwartungskonform	Attraktivität
2	unverständlich	○	○	○	○	○	○	○	verständlich	Klarheit
3	kreativ	○	○	○	○	○	○	○	phantasielos	Neuartigkeit
4	leicht zu lernen	○	○	○	○	○	○	○	schwer zu lernen	Klarheit
5	wertvoll	○	○	○	○	○	○	○	minderwertig	Stimulation
6	langweilig	○	○	○	○	○	○	○	spannend	Stimulation
7	uninteressant	○	○	○	○	○	○	○	interessant	Stimulation
8	unberechenbar	○	○	○	○	○	○	○	voraussagbar	Zuverlässigkeit
9	schnell	○	○	○	○	○	○	○	langsam	Effizienz
10	originell	○	○	○	○	○	○	○	konventionell	Neuartigkeit
11	behindernd	○	○	○	○	○	○	○	unterstützend	Zuverlässigkeit
12	gut	○	○	○	○	○	○	○	schlecht	Attraktivität
13	kompliziert	○	○	○	○	○	○	○	einfach	Klarheit
14	abstoßend	○	○	○	○	○	○	○	anziehend	Attraktivität
15	herkömmlich	○	○	○	○	○	○	○	neuartig	Neuartigkeit
16	unangenehm	○	○	○	○	○	○	○	angenehm	Attraktivität
17	sicher	○	○	○	○	○	○	○	unsicher	Zuverlässigkeit
18	aktivierend	○	○	○	○	○	○	○	einschläfernd	Stimulation
19	erwartungskonform	○	○	○	○	○	○	○	nicht erwartungskonform	Zuverlässigkeit
20	ineffizient	○	○	○	○	○	○	○	effizient	Effizienz
21	übersichtlich	○	○	○	○	○	○	○	verwirrend	Klarheit
22	unpragmatisch	○	○	○	○	○	○	○	pragmatisch	Effizienz
23	aufgeräumt	○	○	○	○	○	○	○	überladen	Effizienz
24	attraktiv	○	○	○	○	○	○	○	unattraktiv	Attraktivität
25	sympathisch	○	○	○	○	○	○	○	unsympathisch	Attraktivität
26	konservativ	○	○	○	○	○	○	○	innovativ	Neuartigkeit

Zur Auswertung der Fragebögen bietet sich die Nutzung eines Auswertungswerkzeug auf Basis von Microsoft Excel an, welches durch die Autoren unter [STH20a-ol] zum Download angeboten wird. Dort können zunächst zu einem Produkt die Ergebnisse (mehrerer) Fragebögen für jedes Item eingeben werden. Das Werkzeug gibt anschließend die Werte für die sechs Skalen aus, was sowohl für jeden Probanden als auch für Skalenmittelwerte als aggregiertes Ergebnis über alle Probanden möglich ist. Zudem erlaubt das Werkzeug die Ausgabe statistischer Kenngrößen, z.B. des 5% Konfidenzintervalls der Skalenmittelwerte. Die Skalenmittelwerte können entsprechend in das Modell als Attribute einer Evaluierungsbeziehung zwischen einer Softwarekomponente und einem Anwendungsfall übertragen werden.

A4 Übersetzungen aus dem Englischen

In den nachfolgenden Tabellen werden für ausgewählte Ansätze, deren Quellen in englischer Originalsprache vorliegen und für deren Inhalte keine gängigen deutschen Begriffe bekannt sind, die Übersetzungen aufgelistet.

Model-Based Process Engineering nach HUTH et al.

Produktmodell	Product System Model
Prozessmodell	Process Model
IT-System Modell	IT System Model

The Open Group Architecture Framework (TOGAF)

Vorbereitung	Preliminary Phase
Architekturvision	Architecture Vision
Geschäftsarchitektur	Business Architecture
Informationssystemarchitekturen	Information Systems Architectures
Technologiearchitektur	Technology Architecture
Chancen und Lösungen	Opportunities and Solutions
Migrationsplanung	Migration Planning
Implementierungssteuerung	Implementation Governance
Architektur-Änderungsmanagement	Architecture Change Management
Anforderungsmanagement	Requirements Management

Modellbasierte Spezifikation einer MBSE-Modellierungsumgebung nach FRIEDLAND ET AL.

Arbeitsprozesse	Work Process
Aufgabeneinheit	Unit Task
Informationsmodell	Information Model

Beschreibung von Architekturen nach ISO/IEC/IEEE 42010

Architekturbeschreibung	Architecture Description
Architekturbeschreibungssprache	Architecture Description Language
Architektur-Rahmenwerk	Architecture Framework
Architekturbegründung	Architecture Rationale
Architekturmodell	Architecture Model
Modellart	Model Kind
Sicht	View
Sichtweise	Viewpoint

ArchiMate Modellierungssprache für Enterprise Architekturen

Treiber	Driver
Bewertung	Assessment
Ziel	Goal
Ergebnis	Outcome
Prinzip	Principle
Anforderung	Requirement
Einschränkung	Constraint
Bedeutung	Meaning
Wert	Value
Ressource	Resource
Fähigkeit	Capability
Wertstrom	Value Stream
Akteur	Business Actor
Rolle	Business Role
Kollaboration	Business Collaboration
Schnittstelle	Business Interface
Geschäftsprozess	Business Process
Geschäftsfunktion	Business Function
Geschäftsinteraktion	Business Interaction
Geschäftsereignis	Business Event
Geschäftsdienstleistung	Business Service
Geschäftsobjekt	Business Object
Vertrag	Contract
Repräsentation	Representation
Applikationskomponente	Application Component
Applikationskollaboration	Application Collaboration
Applikationsschnittstelle	Application Interface
Applikationsfunktion	Application Function
Applikationsinteraktion	Application Interaction
Applikationsprozess	Application Process

Applikationsereignis	Application Event
Applikationsservice	Application Service
Datenobjekt	Data Object
Knoten	Node
Gerät	Device
Systemsoftware	System Software
Technologiekollaboration	Technology Collaboration
Technologieschnittstelle	Technology Interface
Pfad	Path
Kommunikationsschnittstelle	Communication Network
Technologiefunktion	Technology Function
Technologieprozess	Technology Process
Technologieinteraktion	Technology Interaction
Technologieereignis	Technology Event
Technologiedienstleistung	Technology Service
Artefakt	Artifact
Anlage	Equipment
Werk	Facility
Verteilnetzwerk	Distribution Network
Material	Material
Arbeitspaket	Work Package
Deliverable	Deliverable
Implementierungsereignis	Implementation Event
Plateau	Plateau
Lücke	Gap

Software & Systems Process Engineering Metamodel (SPEM)

Kernelemente	Core
Typ	Kind
Methodenbeschreibungen	Method Content
Aufgabendefinition	Task Definition
Schritt	Step
Artefakt	Work Product
Rolle	Role
Werkzeuge	Tool Definition
Prozessstrukturen	Process Structure
Aktivität	Activity
Spezifische Rollen	Role Use
Spezifische Artefakte	Work Product Use
Prozessverhalten	Process Behavior
Prozess- und Methodenkombination	Process with Methods

Spezifische Aufgaben	Task Use
Methodenpaket	Method Content Package
Prozesspaket	Process Package
Dokumentation	Managed Content
Kategorie	Category
Leitfaden	Guidance
Metrik	Metric
Bibliothek	Method Plugin
Methodenmodul	Method Module
Methodenkonfiguration	Method Configuration
Methodenbibliothek	Method Library

Technische Systeme und ihr Umfeld verändern sich grundlegend und ihre Komplexität steigt. Das macht es notwendig, das Engineering dieser Systeme daran anzupassen. Ansätze wie das Model-based Systems Engineering setzen in vielen Unternehmen umfassende Änderungen bei den Prozessen, Methoden, Modellen und IT-Werkzeugen im Engineering voraus. Die Prozesse, Methoden und Modelle müssen dazu häufig zusammenhängend betrachtet werden, um daraus konsistente Anforderungen an die IT-Architektur abzuleiten. Für diese ergibt sich oftmals ein komplexer Lösungsraum mit einer Vielzahl von IT-Werkzeugen und Schnittstellen, wobei die Anforderungserfüllung der verschiedenen Lösungsvarianten sehr unterschiedlich sein kann. Dies erschwert aufgrund vieler Abhängigkeiten nicht nur die Bildung der Lösungsvarianten, sondern auch deren Bewertung. Ganzheitliche und zusammenhängende Modelle der Prozesse, Methoden, Modelle und IT-Werkzeuge des Engineerings können helfen, diese Aufgaben systematisch und effektiv zu unterstützen.

Ziel dieser Arbeit ist eine Spezifikationstechnik zur Analyse, Gestaltung und Bewertung von Engineering-IT-Architekturen. Im Kern besteht diese aus einem Sprachkonzept, welches Sichten, eine Modellierungssprache und die Vernetzung der Modellelemente definiert.

Ein Vorgehensmodell leitet den Anwender an und erlaubt in Kombination mit einer Werkzeugunterstützung die praktische Anwendung. Ein industrielles Beispiel aus der Landtechnik demonstriert den Einsatz der Spezifikationstechnik.