Lehrstuhl für Produktentwicklung
der Technischen Universität München

A Method for Product Architecture Management

In Early Phases of Product Development

Frank R. Deubzer

Vollständiger Abdruck der von der Fakultät für Maschinenwesen
der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. habil. Boris Lohmann
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. Udo Lindemann
	2.	Prof. Tyson R. Browning, Ph.D., Texas Christian University, Fort Worth, USA

Die Dissertation wurde am 22.12.2014 bei der Technischen Universität München
eingereicht und durch die Fakultät für Maschinenwesen
am 24.06.2015 angenommen.

FOREWORD OF THE EDITOR

Problem

The paradigm of mass production is being continuously challenged due to the growing relevance of niche markets, saturated unpredictable markets and shorter use phases and lifecycles of products. The resulting challenges for the development of products show before all in the shortened development cycles, continuous need for innovations and increased product complexity. Different strategies emerged to meet these requirements. While business and production strategies such as mass customization address the later phases of the product lifecycle, research in engineering design and systems engineering provide promising and applicable methods and approaches for the early phases of the product lifecycle. The product architecture plays an important role in the early phases of product development: as the result of the business process to fulfill the requirements of the market. Since existing approaches address different aspects and steps of the process, a comprehensive procedure and model are necessary, to allow for the integrated application of different methods and models along the iterative and recursive design process.

Objectives

With respect to the described challenges, the presented work has to provide an entity framework of the product architecture, underpinned by a suitable modeling approach. The framework must further contain suitable methods and approaches to support the early phases of product architecture development. To enable the implementation of methods and approaches proposed, a procedural model needs to be introduced, capable to support the system architect comprehensively during the development process. Given the dynamic and manifold requirements, the approach is required to be consistent, comprehensive and flexible. To achieve these goals, an intensive discussion of the role of product architecture and the existing approaches and methods to cope with the product architecture is required. In addition, the area and understanding of complexity management should give valuable insights. To complete the overall approach, identified gaps are to be closed by developed and tested solutions. The underlying scientific approach has to be based on generally accepted scientific qualities and include the reasonable application of descriptive and prescriptive studies.

Results

The presented work provides a comprehensive overview on the role of product architecture and means to cope with complexity in the context of engineering design. With the conducted method review, different schools of thought and fields of research are characterized and their suitability for product architecture management in the early phases analyzed. Based on the discussion, appropriate conclusions are drawn and missing constituents developed, such as the coupling of methods and models and the coping with recursive procedures, integrating analysis and synthesis during the design phase. The presented research results in three main outcomes: the architecture entity framework, architecture model, and procedure for architecture management. The architecture framework enables a comprehensive situation analysis, start into the architecture project and structuring of architecture information. The results of each activity of the procedural model provide a substantial part of information to the overall picture, depicted in the architecture model accessible for involved stakeholders. The case study-based example gives a practical insight of the application of the approach for the management of product architectures. The procedural steps and underlying methods of the approach are validated within the example. The overall approach and combination of framework, model, and procedural model prove to be feasible. Especially the coupling of different methods for analysis, synthesis and depiction of solution space are conducted systematically based on the defined procedure for architecture management.

Conclusions for industrial applications

For system architects in industry, the framework alone gives insights into which product architecture entities to consider and how they are related, both within and between the entities' domains. The procedure and provided methods can be applied in projects of different nature, for analysis or synthesis for example, due to the flexibility of the approach. Since the presented research results in an integrative approach supporting recursive and iterative processes, common methods in industrial application are increased in value. Their results can be reused and are integrated in preceding and subsequent processes. The overall approach turns out to be comprehensive yet pragmatic and flexible in its application to suit different situations and challenges in industry.

Conclusions for scientific research

Researchers of related fields may find the discussions on product architectures, coping with complexity, and insights on potential methods for the managing of product architectures of major interest. The mentioned fields of research are discussed in sufficient depth and set into a consistent scientific context and interrelated with each other. The focus on product architectures is giving the discussion a fresh momentum and is linking methods together in a novel fashion. The additionally developed methods provide a sound supplementation of research in Systems Engineering and Multiple Domain Modeling. The challenges in industrial application and scientific gaps are coherently deduced and result in accurate conclusions. The framework and procedure are generally applicable and comprehensive, allowing following researchers to build upon the results and continue the presented trains of thought.

Garching, June 2016

Prof. Dr.-Ing. Udo Lindemann

Institute of Product Development
Technische Universität München

ACKNOWLEDGEMENT

This work is based on my research conducted at the Institute of Product Development at the Technische Universität München from September 2004 to April 2010. I would like to thank Prof. Dr.-Ing. Udo Lindemann for providing a research environment based on trust, freedom, and encouragement, enabling me to develop ideas and research in different areas and directions. The possibility to work with academic and industrial partners both nationally and internationally considerably enriched the presented work. With continuous belief and patience, Professor Lindemann helped to overcome one or the other inevitable dry spell. Additionally, I would like to thank Professor Tyson Browning for his valuable input on the thesis and for widening the view onto the presented problem based on his experience; and Professor Boris Lohmann, who acted as chairperson of the examination board and managed the dissertation process fast and smoothly.

Very important for my work was the collaboration with the colleagues at the institute, who not only were great to work with, but with whom lasting friendships emerged. I would like to thank my friend Matthias Kreimeyer for the numerous shared hours in and outside the institute, due to the many projects and publications we worked on together. The intensive input on the research and the proof reading of my dissertation are only two examples for which I am thankful. This applies accordingly to the friends and colleagues whom I could not share my full research-time with: Ulrich Herfeld, Thomas Braun, Mike Maurer, Christoph Baumberger, Josef Ponn, and Andreas Gahr supported my research at the institute at any time and gave valuable input in common projects and publications. I have been blessed to share the office with such great colleagues as Luc Felgen, Frank Hoisl and Stefan Langer. I am grateful for the time together and fortunate to have gained such close friends. The same goes for my friends Martin Graebsch, Clemens Hepperle, and Holger Diehl.

I would also like to thank the personnel at the institute for their support, and of course the numerous students, whose work has contributed to this thesis. Armin Förg, representative for many others, is worth mentioning due to his valuable contribution to different projects over the years.

In the end, it is my family to whom I want to express my deepest gratitude: to my parents, not only for the education I was offered, but also for the support over the years in both my personal and professional life, and to my sister and her family for the support and comprehension over many years.

Above all the aforementioned, I want to thank my wife Veronika. She has been the most patient, supporting, and understanding person I know. Her character, affection, and energy backed me up during all the stressful and downcast days and long nights. I am truly blessed with her and our children.

Munich, June 2016 Frank Deubzer

CONTENTS

Acronyms

ADT	Axiomatic Design Theory
AF	Architecture Frameworks
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance
CAD	Computer-Aided Design
CADM	C4ISR Architecture Data Model
CAE	Computer-Aided Engineering
CPM	Characteristics-Properties-Modeling
DfX	Design for X
DMM	Domain mapping Matrix
FBS	Function-Behavior-Structure
FMEA	Failure Mode and Effect Analysis
HIL	Hardware in Loop
IAF	Integrated Architecture Framework
IDEF0	Integration Definition for Function Modeling
MDM	Multiple Domain Modeling
MoDAF	Ministry of Defense Architecture Framework
OOA	Object-Oriented Analysis
QFD	Quality Function Deployment
SADT	Structured Analysis and Design Technique
SIL	Software in Loop
SysML	Systems Modeling Language
UML	Unified Modeling Language
VDI	Association of German Engineers [Verein Deutscher Ingenieure]

1. Introduction

The role of product architecture in the manufacturing industry, as well as in engineering design research and other research areas such as business management, is diverse. Different concepts and perceptions exist in both worlds, underlining the versatile nature of product architectures and the many perspectives that can be taken to understand, to describe and, finally, to be able to "manage" product architectures. The "Management of Product Architectures" in the context of this work will be intensively discussed in the following sections and can be briefly outlined as the coping with the complexity of the product architecture itself and with the complexity resulting from its interrelations with the various interacting surroundings. The surroundings are mainly influenced by the interconnectedness of the product architecture with the company organization structure and the process architecture, as well as the challenging and dynamic economic environment. Given the many different phases of a product's lifecycle and the numerous different classes of artifacts of product architecture, the complexity is clear. Different methods, methodologies, and theories have emerged, due to these circumstances. This first chapter will clarify not only the underlying terminology of this work, but also point out the recurring challenges and the resulting objectives, despite the existing efforts and solutions.

1.1 The role of product architecture in engineering design

It would be a considerable understatement to report that the challenges for engineering design in industry are underlying change over time. In fact, the dynamic changes in both corporate financing and real economy, as well as the worldwide developments in society, legislation and ecology all serve to increase and multiply the demands for all corporate activities. For decades, the challenge for engineering design in industry was described as the area of conflict between the three major co-dependent issues of time, cost (or productivity) and quality [BROWNING 1998, pp. 260 ff., CLARK & FUJIMOTO 1991, p. 70, EHRLENSPIEL et al. 2007, p. 21, LAWSON & KARANDIKAR 1994, WILDEMANN 1999, p. 18], resulting in compromises for the company, customer or product.[1] Efforts in all three directions manifested in research approaches and methods or methodologies, some of which have, by now, become standards in industry. Examples for these efforts are the numerous approaches for the optimization of workflows and business processes [CLARKSON & HAMILTON 2000, EPPINGER et al. 1997, KREIMEYER 2010, WYNN 2007] aiming for the understanding of causes for the origin of delays and iterations[2] in processes, as well as measures for their identification and prevention.

[1] The challenges of "time, cost and quality" do not apply exclusively to development, but can also be found in business processes, manufacturing and other areas of the manufacturing firm as well. The presented work focuses on the challenges related to the development of technical products.

[2] The reduction of delays and iterations in processes can only serve as an example for the optimization of processes. The term "waste" (originating from the lean production philosophy) is increasingly applied, not only for the processes in manufacturing, but in development as well. On the basis of this definition, the field of

The reduction of cost, aside from being a side effect of the reduction of time in business processes, produced different efforts in understanding the origin of cost and the economical design of products [Zirkler 2010, Ehrlenspiel et al. 2007]. Efforts to increase quality in product development are most visible in industry, whereas the application of the methods "Quality Function Deployment" (QFD) and "Failure Modes and Effects Analysis" (FMEA) is very common and, in the case of FMEA, a substantial part of the development process and the Supplier-OEM (Original Equipment Manufacturer) relationship.[3] Quality efforts exceed the field of product development and can be found in process management and manufacturing, as well as other fields.

The major issues of time, cost and quality can only provide the coarse boundary conditions for engineering design. The Design for X (DfX) methodology [see e.g. Huang 1996, Lindemann 2007, Weber 2007] embraces numerous other issues to be considered in engineering design. The different "X" or "aspects" [Lindemann 2007] of Design for X stand for different design purposes. Design for Manufacturing or Design for Assembly are considered to be the most investigated DfX aspects [Boothroyd et al. 2002], while Pahl et al. [Pahl et al. 2007, p. 234 and pp. 308ff.] present for example ergonomics, recycling, maintenance, safety and others as further aspects.[4] The process of dealing with the various DfX aspects is still described as "chaotic" [Lindemann 2007]. The interdependencies between different aspects are one cause. Research is ongoing regarding how to cope with the different existing guidelines and rules of each single aspect and their interdependencies [see for example Bauer & Meerkamm 2007 or Weber 2007].

As dynamic changes drive the different competing aspects of engineering design to new levels, the emerging complexity turns out to be the main factor summarizing the challenges in engineering design. Thus, research in different areas about the management of complexity has reached a new high. Systems engineering and Systems Theory gained importance in recent years and, according to reviews in industry, will be increasingly important in the following years [Bullinger et al. 2003, Deubzer et al. 2005]. The coping with complexity is conducted from different perspectives, such as the human perception of complex systems [Dörner 1992] as well as the management [Clarkson et al. 2004, Eppinger & Salminen 2001, Schuh 2005, Wildemann 1999] or structural considerations of complex systems [Browning 2001, Maurer 2007].

For the area of product development, the market, organization, process and product were defined as the four relevant fields where complexity occurs and emerges [Maurer 2007, p. 3]. Lindemann et al. take a similar position, stating that products and processes have become

process optimization far exceeds the reduction of delay and iterations, but aims for profound management and goal orientation ("customer value") of processes.

[3] FMEA is a substantial part of the contract between the OEM and the supplier as a required service to be performed by the supplier.

[4] Huang describes a pattern to define and differentiate the aspects by the equation "X = x + bility", where "x" stands for a certain lifecycle business process while "bility" is to be replaced by performance measures, for example "Design for Assembly Cost" [Huang 1996, p. 3].

more and more complex, due to the increase and diversification of customer needs in the buyers' markets [LINDEMANN et al. 2006, p. 1, PINE 1993, p. 31]. A similar conclusion is reached in business management by DANILOVIC & SANDKULL, who differentiate between people, technology and functionality as sources of complexity and transfer those into the aspects of product, organization and process as "basic dimensions" of the management of complexity, in which the market aspects are inherent in the functionality demands placed on the product and by the process [DANILOVIC & SANDKULL 2004]. EPPINGER comes to the same conclusion, describing product, process and organization as the three relevant interrelated domains in product development [EPPINGER 2001]. WILDEMANN describes complexity from a business management perspective as an issue in requirements, technology, product and development methods [WILDEMANN 1999, p. 11]. YASSINE & WISSMANN add the knowledge portfolio and production and distribution on the company side, as well as the brand portfolio and marketing on the consumer (or market) side [YASSINE & WISSMANN 2007]. DANILOVIC & BROWNING set the product architecture into the context of the following aspects: process and organization (in the following companies and organizational structure), goals (in this work considered within the chapter "Markets and requirements"), and tools [DANILOVIC & BROWNING 2007].[5]

Here, the product architecture is understood as the result of the business process, performed by the organization and successfully fulfilling the requirements of the market. Based on this understanding, the following chapters will clarify the implications of the three fields on product architectures. The implications will point out the need for action in the context of product architecture management and lead to the discussion of a precise understanding of product architectures in industry and research.

1.1.1 Markets and requirements

As the first field of impact on the product architecture's complexity, the market provokes the most critical implications, dominating other origins of complexity [MAURER 2007, p. 4]. The dynamic developments of the market underlie a continuous change, which can be roughly characterized as an interaction of the technical possibilities in development, technology and production, on the one hand, and the accordingly increasing customer requests on the other [MCKENNA 2000, p. 17, PINE 1993, p. 27, WILDEMANN 1999, p. 62]. This interplay is characterized as the feedback loop of mass production (see Figure 1-1) by PINE for the period of the mid 20th century [PINE 1993, pp. 25-32].

The underlying assumption is therefore that new products are produced in such a quantity that mass production processes can be implemented to produce standardized products with reasonable quality and costs. These products are introduced to a homogenous market with

[5] The extensive area of tools, albeit highly relevant especially for the practical application of methodical approaches, will not be considered as separate section in this work, yet the implications will be considered when relevant.

stable demands[6] and thus lead to long product lifecycles with accordingly long product development cycles. Until the 1970s, the elements of the feedback loop were reinforcing one another, leading to the highest possible volumes and most efficient processes in manufacturing, serving only the largest possible markets with products [PINE 1993, p. 27].

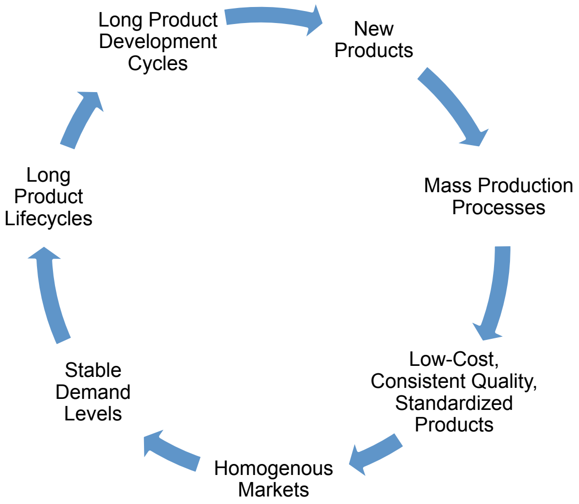

Long Product Development Cycles
New Products
Mass Production Processes
Low-Cost, Consistent Quality, Standardized Products
Homogenous Markets
Stable Demand Levels
Long Product Lifecycles

Figure 1-1 Paradigm of Mass Production [PINE 1993, p. 27]

Market dynamics increased over the past years, as different developments caused changes to multiple elements within this circle, so that the described reinforcing characteristic is no longer a valid model of reality. After the high of mass production, market saturation was reached in the identified homogenous markets [LÖSCH 2001, p. 32]. The "typical" customer, being the largest market share, was replaced by a number of niches, which became more important to further increase the sale of products.[7] Markets have become more heterogeneous, diverse and unpredictable [MCKENNA 2000, p. 23]. Causes of the diversity of markets are the growing and diverse customer demands, accompanied by increasing quality standards and international competitors, due to globally distributed markets and know-how [LÖSCH 2001, p. 32]. More individualized products are required for the resulting niche markets with "similar" products, but differentiated customer specifications [PINE 1993, FRICKE & SCHULZ 2005]. Individualization in terms of tailored products was closely related to craftsmanship and consequently higher prices during times of mass production. Customers today demand products of increasing quality and reasonable prices that closely match their individual needs [PINE 1993, pp. 45-46, MCKENNA 2000, pp. 17-18]. This customer behavior is intensified by the access to information and knowledge about worldwide offers and the changing social standards and values [LÖSCH 2001, p. 32-34]. According to PINE, the paradigm of mass

[6] The stabilization of homogenous markets is in that case achieved by ignoring niche markets [PINE 1993, p. 27].

[7] The continuous growth of sales and thus markets was one of the major principles of mass production to keep the established systems profitable [PINE 1993, p. 27].

customization represents the inversion of the paradigm of mass production. Instead of the approach of unified markets, long lifecycles and standardized products, niche markets and shorter lifecycles demand customized products [PINE 1993, p. 48].[8]

The tailoring of products to market niches was achieved in postproduction and aftersales during the period of mass production. Due to the cost of aftersales services, the intention of the philosophy of mass customization was to shift the tailoring of products upstream. As a result, production has to provide increased variety, which evoked different approaches in manufacturing to enable flexibility in manufacturing and assembly [see e.g. DE LIT & DELCHAMBRE 2003]. Requirements for flexible production are more expensive and capable machinery, new technologies and highly skilled workers [PINE 1993, p. 46]. Until today, the concept of mass customization could not yet be fully implemented, especially when technical products are involved. The complexity of technical products and their production could not yet be handled to produce individualized and customized products economically.[9]

As niche markets are smaller and changing both constantly and rapidly, the increasing variety has to be achieved more rapidly. A dramatic reduction of product development times is required, as product lifecycles also shorten accordingly. Shortened lifecycles are accompanied by the continuous improvement of products and the replacement and improvement of technologies, both in products and in production [PINE 1993, p. 46]. Currently, the half-life of technologies tends to be even shorter than the lifecycle of products, increasing the pressure on the length of development cycles [FRICKE & SCHULZ 2005].[10] The customer demands, on the other hand, are growing with the technological possibilities [ENGEL & BROWNING 2008].

To maintain this reinforcing circle, companies are required to provide continuous innovation in their products. Customer demands are one driving factor; others include the field of worldwide competitors [LÖSCH 2001, p. 32] or the threat of product piracy due to known and manageable technologies [WILDEMANN et al. 2007]. Innovative solutions more often turn out to be incremental, rather than radical or breakthrough[11] innovations, although continuous incremental innovation can lead to breakthrough innovations [PINE 1993, p. 114]. The

[8] The relevance of niche markets for the producing companies is documented in numerous works, such as “Markets of One” [GILMORE & PINE 2000], “The Long Tail Phenomenon” [ANDERSON 2008], or the expression “selling big by selling small” [MCKENNA 2000, p. 24] .

[9] PILLER provides examples of the implementation of mass customization and discusses failed attempts. The presented cases, successful or not, deal with individualized products which are rarely technical. The approaches of mass customization consider mainly the economical and customer relationship point of view, rather than the development and production of technical products. Examples stem primarily from clothing, food, furniture, luxury, or service industries (such as airlines or web-based products) [PILLER, 2001, pp. 393-409, PILLER & STOTKO 2003].

[10] FRICKE & SCHULZ identify the integration of electronics and software - both with very short half-times - into products as a major cause of this gap [FRICKE & SCHULZ 2005].

[11] The terms “radical innovations“ and “breakthrough innovations” will be used equivalently throughout this work.

coherence of the need for innovations and the limited available development time increases this trend, so that radical innovations[12] are less likely in the prevailing environment, leading to the conflict between the demand for radical innovations, on one hand, and the lack of ability to develop radical innovations, due to the given constraints, on the other.

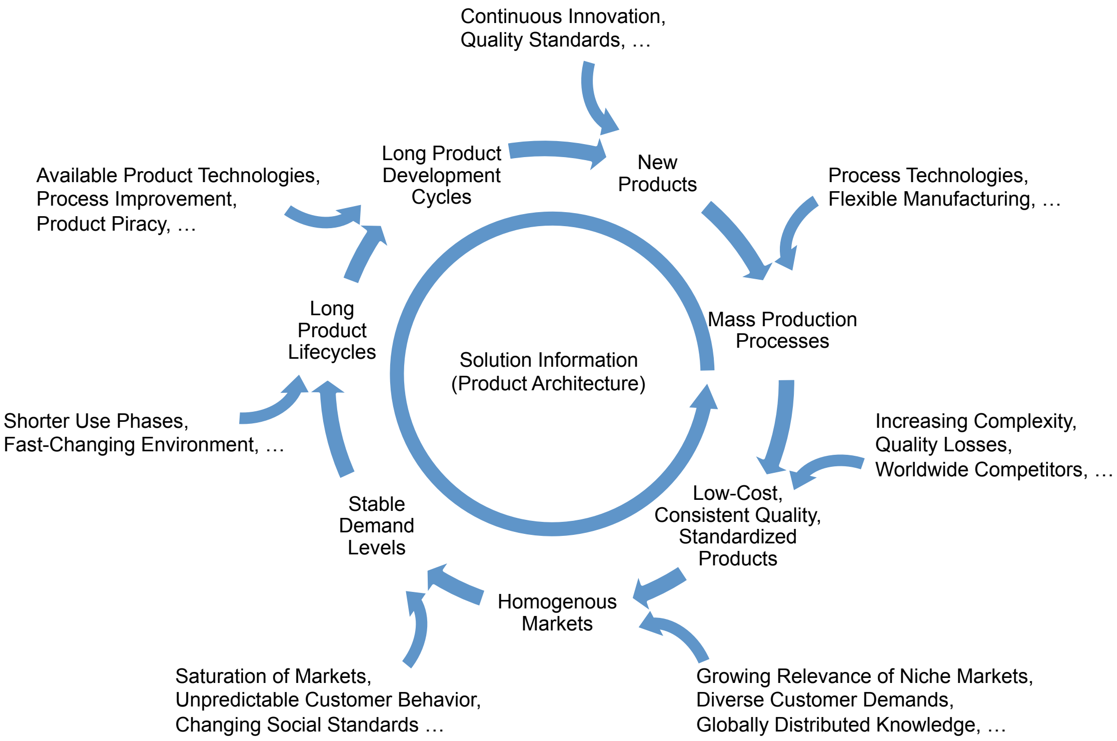

Continuous Innovation,
Quality Standards, …
Long Product Development Cycles
New Products
Available Product Technologies,
Process Improvement,
Product Piracy, …
Process Technologies,
Flexible Manufacturing, …
Long Product Lifecycles
Mass Production Processes
Solution Information (Product Architecture)
Shorter Use Phases,
Fast-Changing Environment, …
Increasing Complexity,
Quality Losses,
Worldwide Competitors, …
Stable Demand Levels
Low-Cost, Consistent Quality, Standardized Products
Homogenous Markets
Saturation of Markets,
Unpredictable Customer Behavior,
Changing Social Standards …
Growing Relevance of Niche Markets,
Diverse Customer Demands,
Globally Distributed Knowledge, …

Figure 1-2 Recent implications on the paradigm of mass production and mass customization

The implications of the market as a driver of complexity can be summed up as the growing relevance of niche markets, saturated unpredictable markets, shorter use phases and lifecycles. The relevance for development and manufacturing shows in the shortened development cycles, continuous innovations in products, required flexibility in manufacturing and increased product complexity and quality losses. Figure 1-2 shows some of these implications, on the basis of the introduced paradigm of mass production (clockwise). The counterclockwise arrow depicts the paradigm of mass customization, introduced by PINE in the early 90s [PINE 1993, p. 45]. The two general principles of mass customization i.e. close customer relationships and moving the customization of products further upstream, seem to be valid reactions to dynamic markets. With mass customization being a business strategy, those two components are necessary but not sufficient for a successful application. The information-intensive proceeding must be supported by a management and depiction of information flows in product design, such as the solution space [PILLER et al. 2004]. The

[12] A radical innovation requires both a changed linkage between system elements (architectural innovation) and overturned core concepts according to HENDERSON & CLARK 1990 [for a detailed discussion and differentiation also see HENDERSON & CLARK 1990, p. 12].

missing link when considering the manufacturing and marketing of customized products turns out to be the product architecture and valuable knowledge about its variants, alternatives, functions, requirements etc. The product architecture thus moves into the center of the existing paradigms in Figure 1-2 as a central enabler and necessary for the provision of customized, variant-rich products.

Based on the simplified model of a product lifecycle, Figure 1-3 depicts the upstream shift of product customization relevant for a successful offering of products for niche markets, based on a reduced product lifecycle model.[13] During mass production, customization according to customer demands was achieved throughout aftersales and service, focusing on the distribution, utilization and maintenance of the product. The mass production paradigm left the customization untouched during production to achieve the explained cost and mass effects. The business model of mass customization shifted the customization of products to flexible manufacturing systems and emphasized the relevance of customer relationships and the integration of customer needs into production and other business processes. The relevance of customization during product planning, development and design as a key factor for successful product customization is emphasized by the outstanding position at the beginning of the product lifecycle. Extensive research was conducted for the design and development of variant-rich products, due to these circumstances. Different authors provide an overview on the current state of existing methods and approaches [see for example FIXSON 2007, JIAO et al. 2006, or RENNER 2007, pp. 49-90]. A detailed discussion of the different methods and techniques and the relevant phases of the product lifecycle to which they apply will be conducted in the later chapters of this work.

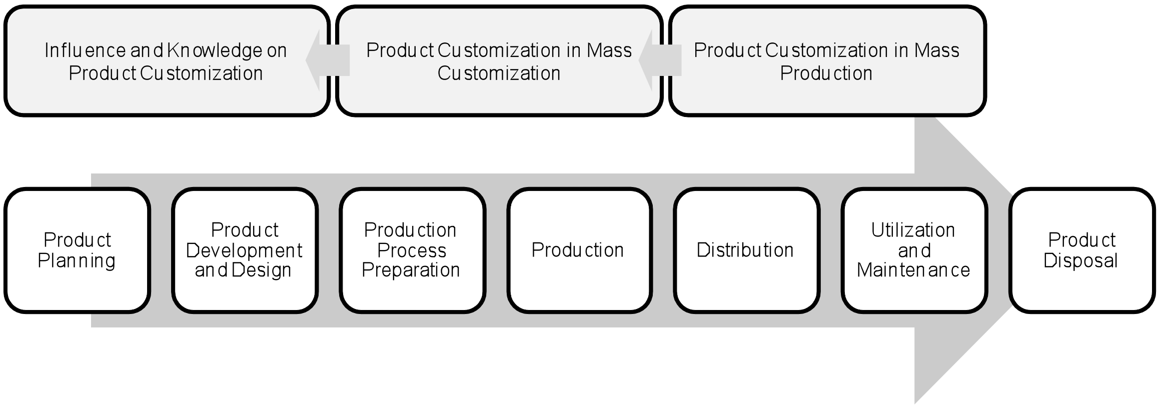

Influence and Knowledge on Product Customization
Product Customization in Mass Customization
Product Customization in Mass Production
Product Planning
Product Development and Design
Production Process Preparation
Production
Distribution
Utilization and Maintenance
Product Disposal

Figure 1-3 Upstream shift of product customization in the product lifecycle

With the customer requirements being the main focus[14] of activities of the producing industry and engineering design, the different Design for X aspects state even more requirements and

[13] The depicted phases of the product lifecycle are derived from HEPPERLE et al., who developed an integrated and networked product lifecycle model based on different existing lifecycle models [HEPPERLE et al. 2009a].

[14] EHRLENSPIEL ET AL. describe "function fulfillment" as *the* factor for successful product development, followed directly by achieving the time and cost goals [EHRLENSPIEL et al. 2007, p. 21].

functions to be fulfilled by a product architecture not visible for or knowingly recognized by the customer. LINDEMANN lists a large number of Design for X aspects in alphabetic order, extendable at will. Among them are manufacturing and assembly, but also different types of cost (lifecycle, operating, general), technical aspects (noise, robustness, tolerances, vibration etc.), functional (cleaning, comfort, ergonomics, safety, usability etc.), strategic (company targets, corporate identity etc.), organizational (logistics, sub suppliers etc.) and other aspects [LINDEMANN 2007]. More tangible, HERFELD gives a practical example, citing GRABNER & NOTHAFT, with requirements to be considered in car body design (e.g. package, design, guidelines, acoustics, corrosion, surfaces, body shell, statutory requirements), as well as the downstream activities of the product lifecycle (e.g. testing, simulation, controlling, assembly, manufacturing) [HERFELD 2007, p. 19, GRABNER & NOTHHAFT 2002, p. 1].[15] CRAWLEY et al. differentiate between “direct” design to achieve the main functions (required immediate for the product’s purpose), and the planning of the lifecycle (posing requirements such as manufacturing, upgrade etc.), plus the so-called “ilities” (reliability, flexibility etc.) [CRAWLEY et al. 2004].

Connecting these findings to the product architecture, the resulting complexity and the challenges in engineering design are evident. This complexity requires a reasonable handling of variant-rich product architectures with multilayered requirements and numerous existing points of view, stemming both from the market, as well as from the comprehensive lifecycle perspective.

1.1.2 Companies and organizational structure

The complexity of the organization has gained an important impact, both in terms of the organizational structure and the process architecture [LAWSON & KARANDIKAR 1994]. Organizational complexity has become one of the key challenges in profitable economic activities [EPPINGER 2001]. Markets and requirements, as was argued in the previous chapter, pose a significant cause for the companies’ internal complexity. The shortened lifecycles and development cycles, as well as the broadening of the product portfolio, both on the market as well as in manufacturing, evoke and require an equally dynamic and multi-faceted organization. The resulting division of labor in concurrent engineering[16] processes causes difficulties in establishing an efficient organizational structure, as well as in the process architecture. Two different aspects are largely relevant in this context: the establishment and cooperation of multidisciplinary teams [SOSA et al. 2004] and the parallel or multiple project environments [LAWSON & KARANDIKAR 1994]. Further aspects include the spreading of knowledge across the company and, in the case of globally operating companies, spread

[15] HEMEL & KELDMANN propose a reasonable yet simple categorization of Design for X aspects into “virtues” (such as cost, efficiency, quality etc.) and “life phases” (e.g. assembly, distribution, production etc.), in the given example referred to as “requirements” and “downstream activities of the product lifecycle” [HEMEL & KELDMANN 1996, p. 73, compare also CRAWLEY et al. 2004].

[16] Concurrent Engineering is in this context interchangeable with Simultaneous Engineering [PAHL et al. 2007, p. 139].

across the world. This invokes the complexity in making decisions from an organizational view regarding the information, the question of cost efficiency and cost-related knowledge and the company surroundings, in terms of suppliers, customers, competitors, legislation etc.

The task of team organization is closely related to the character and complexity of the product architecture when discussing the context of product design and development [SOSA et al. 2004]. After the definition of processes, workflows and related tasks, the difficulties of team organization show in the assignment of tasks to the right persons. In general, the assignment can be supported through the definition of different models of tasks, roles and the interrelation of the two. To give an example, CHEN derived a team member model, based on a comprehensive literature review, incorporating the functional expertise, teamwork experience, communication skills, the flexibility in job assignment and personality traits [CHEN 2005]. In engineering design, different authors come to the conclusion that the product architecture is the key communication catalyst among people involved in the design process. BRADLEY & YASSINE point out the relevance of the composition of design teams, based on the product architecture characteristic. When comparing the team structures and product architecture, the different clusters in both structures match for more than 60 % in the presented use case. As a conclusion, BRADLEY & YASSINE state that the efficiency of communication is equal to that value, probably resulting in issues during the design process, due to lack of communication channels resulting from the mismatch of more than 30 % between organizational and product structure [BRADLEY & YASSINE 2006]. KREIMEYER et al. come to the same conclusion when analyzing the collaboration between engineers in design and simulation (Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE)), stating that more than 60 % of engineers do not have access to the information required for their task, largely resulting in poor results within their work [KREIMEYER et al. 2006]. As a solution to this problem, HERFELD introduced an approach for efficient collaboration based on the derivation of teams according to the product architecture [HERFELD 2007, pp. 154-171]. The concept of interdependency between product architecture and organizational structure is also stressed by [GÖPFERT 1998]. KUNZ et al. underline these findings, stating that, according to their practical experience, efficiency problems during the introduction of new technologies in the automotive industry occur due to organizational, rather than technical issues [KUNZ et al. 1996]. SOSA et al. base their findings similarly on the comparison of team interactions and the interdependencies of the components of the product architecture [SOSA et al. 2004].

The predominant multiple project environments in large companies are further intensifying and complicating the organizational issues of concurrent engineering. DANILOVIC & SANDKULL describe the situation as a multitude of interdependencies between projects, tasks and activities, people and knowledge areas. Furthermore, they refer to the important role of the product architecture in terms of technologies, products, and components [DANILOVIC & SANDKULL 2004]. As a prime example, the automotive industry executes multiple projects in terms of product lines, which are developed by a single manufacturer. Not only are different components of one model designed in parallel, but also the development processes of product lines overlap to allow for a sequential release of different models.[17] Apart from the temporal issues of coordinating the necessary and numerous interrelated tasks, the distribution of knowledge inherited by the involved people and the assignment to teams due to their qualifications are the main causes for difficulties in such an environment, as discussed in the following sections. At any point in time, the current status is different for each project, so that the use of, for example a new technology, might be practically applicable in one project or product line but not suitable for application in others. The reason is primarily the degree of maturity that a product has already reached or, in the opposite case, a very immature state of development. The same is valid for the reuse of the same parts in different product lines. These circumstances evoke the necessity of staff functions to coordinate the carry over, as well as the introduction of new technologies, communication of best practices etc. The briefly described necessities create a more complex organizational environment, due to the multidimensional character of organizations with numerous parallel tasks and projects.

In the context of the organizational structure and multiple project landscapes, information and knowledge were identified as key issues to solve in order to manage projects successfully. Naturally, knowledge and information is distributed across the company [PINE 1993, p. 115], as different tasks and competences are inherited by different departments and persons. Further enlarged by multi-project environments, the implications of spread knowledge are omnipresent [PINE 1993, p. 115]. The volume of knowledge and information required can be estimated using the example of car body design established by HERFELD that was presented in the previous section, indicated the numeroud requirements to be considered during development [HERFELD 2007, p. 19]. Surveys have shown that the search for information consumes a large amount of an engineer's time [KREIMEYER, M. et al. 2006]. For successful development, existing information and knowledge has to be available and connected, as poor availability of information creates a bottleneck in the development of products [EHRLENSPIEL et al. 2007, p. 25]. Product development itself can be described as an information-processing activity [KUSIAK 1999, p. 201]. If not in the design department, knowledge about cost, production technology, customer needs and lifecycles is inherited in different departments, causing deficiencies and disturbing the information flow between involved persons, e.g. due to personal reasons, such as lack of time, conceitedness, shyness [EHRLENSPIEL et al. 2007, p. 25] or the individual goals of people [BLACKENFELT 2001, p. 12]. PINE underlines the important role of the sharing and connection of information from the perspective of value creation and innovation for the product architecture. The close collaboration of departments, for example design and production, can result in immediate profit for the organization in terms of the creation of innovative solutions. At least, according to PINE, continuous (incremental) innovations are more likely to occur in environments where different departments collaborate closely [PINE 1993, p. 15]. In general, knowing, understanding and considering the downstream effects of decisions is a critical aspect of successful engineering,

[17] According to RENNER, the number of derivates offered by the BMW Group grew from 6 in 1985 to 19 in 2005 [RENNER 2007, p. 19]. At the same time, market segmentation in the automotive sector grew from 9 segments in 1987 to 35 segments in 2007 [RENNER 2007, p. 33], according to HERFELD even more than 40 segments existed in 2005 [HERFELD 2007, p. 8].

underlining the role of knowledge and information in understanding organizational complexity; the application of appropriate methods of e.g. data and knowledge management, is also an important aspect [BLACKENFELT 2001, p. 11-12].

The numerous and continuous decisions that are an important element in the design process are closely connected to the characteristics of knowledge and information availability and distribution in companies' organizations [DE BOER 1989, p. 59]. Decisions regarding a product architecture are required at the end of every step of the process, as well as recursively during the steps [ROPOHL 1975, p. 58]. SIMMONS describes systems architecting as a decision-making process [SIMMONS 2008, p. 18]. Considering the complexity of product architectures, the most important decisions include typical performance metrics such as market shares, project success, form and function or efficiency [KRISHNAN & ULRICH 2001].[18] DE BOER points out the available knowledge as one out of four relevant factors for decision processes, based on a questionnaire applied to practitioners [DE BOER 1989, p. 102-104].[19] A similar role of knowledge in decision-making is underlined by HATAMURA [HATAMURA 2006, p. 5]. [20] Depending on the companies' organizational structure, decision-making processes may vary from decisions made by individuals (primarily hierarchical structures) and groups (networked organizational structures) [DONG 1995]. While the predominant organizational structure was hierarchical in the past century [DONG 1995], recent dynamics of markets made networked organizations the prevailing organizational structure. Thus, decision-making processes are, on one hand, based on a greater amount of knowledge, due to the involvement of numerous persons and departments; on the other hand, the decision-making itself becomes more complex, due to the numerous prevailing opinions and goals, entitled a "lack of integration" by BLACKENFELT [BLACKENFELT 2001, p. 12], often resulting in the "sub-optimization", i.e. the optimization concerning one criterion and thus reducing overall performance, resulting in unbalanced and sub-optimal tradeoffs. As was defined, decision-making is based on valuable information. EEKELS defines four classes of information required for a decision: factual information (information about the available alternatives to choose from), normative information (defining the standard against which the factual information has to be compared, such as requirements and needs), intuitive estimation (estimated information not available as factual information; estimations based on experience and related factual information) and methodical information (comparing factual and normative information and rank evaluation

[18] The described metrics are derived from the academic communities marketing, organizations, engineering design and operations management by KRISHNAN & ULRICH, who provide a detailed overview of the different perspectives on decision-making in product development [KRISHNAN & ULRICH 2001].

[19] The complete model of DE BOER includes the personal qualities, fixed characteristics (physical, nature) and external influences (e.g. customers or principals) as properties of a person influencing the decisions. Outside influences include the problem itself and the available facilities [DE BOER 1989, p. 104].

[20] For HATAMURA, experience, preference, hunch and lifestyle are the influences accompanying knowledge in decision-making processes [HATAMURA 2006, p. 5]

outcomes) [ROOZENBURG & EEKELS 1990, p. 8-9].[21] For economic decisions, the cost information in particular (factual and normative) is relevant to continuously compare actual and anticipated cost performance [EHRLENSPIEL et al. 2007, p. 25-26, KRISHNAN & ULRICH 2001]. Methodical information is perceived as equally crucial. Method selection, as well as the support of method application, are very important for successful project execution, especially concerning costs [EHRLENSPIEL et al. 2007, p. 26]. The lack of methods, or the application of wrong methods, are among the most critical variant and complexity drivers in product development [BLACKENFELT 2001, p. 12].

Complex market environments do not only cause complex architectural decisions, but also change the companies' surrounding influences and external dependencies, apart from the market and customers discussed in chapter 1.1.1. As an example, a change of the partnerships with suppliers and the suppliers' role in value networks can be observed [NOVAK & EPPINGER 2001]. In the context of the offering of product variety and mass customization in particular, the role of suppliers has changed in recent years, as their competence and responsibility grows and the systems supplied are accordingly significantly larger [LINDQUIST et al. 2008, compare CLARK & FUJIMOTO 1991, p. 141]. Decision-making and communication (team-building) in this context turn out to be by far more complex than in times before customized and highly integrated products, requiring new methods of management and analysis of company environments [DANILOVIC 2006]. The emerging changes are partly due to a change in thinking and management, such as the Lean thinking movements in response to customer and supplier integration or production [see e.g. WOMACK et al. 2007 and WOMACK & JONES 2006]. Different approaches to solve the difficulties consider the product architecture as highly significant for a comprehensive strategy to achieve successful development projects. For example, LINDQUIST ET AL. identify the modularization of a product architecture as the underlying implication for the modalities and focus of the involvement of suppliers [LINDQUIST et al. 2008]. Accordingly, DANILOVIC focuses on the cross-company team-building on the basis of tasks and work packages based on the product architecture, rather than the functions of business units, to allow for reasonable information flows, transparency and situation visibility [DANILOVIC 2006]. The Lean approach describes the customer needs and the value stream towards their fulfillment as the main focus of economic activity, which in the end leads again to the definition of appropriate products and product architectures [see e.g. GRAEBSCH et al. 2007, WARD 2007, WOMACK et al. 2007].

In consideration of the product architecture, the different aspects of organizational complexity appear to be closely related to its characteristics and properties. Implications of the product architecture on the means of organizations - team building, multiple project environments, knowledge and decision-making, as well as the embedding of a company in value networks - turn out to be of great influence on the success of the companies' economic activities.

[21] Factual and normative information and intuitive estimation form the knowledge required for decision-making itself, while methodic information supports the procedure of decision-making [ROOZENBURG & EEKELS 1990, p. 9]. In this context, SIMMONS differentiates between programmed (routine, can be modeled precisely) and non-programmed (non-routine, models of the system are imprecise) decisions, based on the available information on the problem [SIMMONS 2008, p. 21].

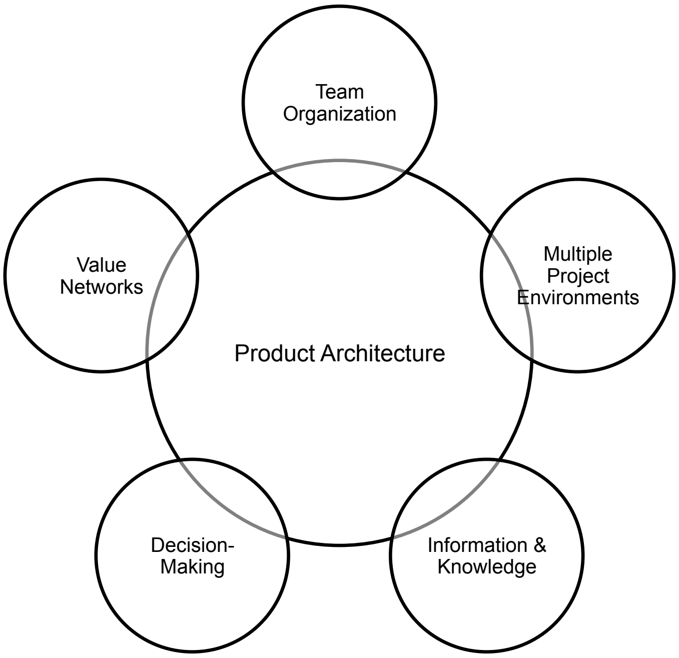


Figure 1-4 Mutual reactions of the product architecture and the organization

A simplified subsumption of the product architecture's implications for the organization is given in Figure 1-4, without further detailing the interdependencies between the architecture and the organizational matters, such as the impact of product modularization and platform strategies on multiple project environments or the existing value networks.

After focusing the discussion on the organizational structure in this chapter, the following chapter will address the relevance of the product architecture for the process architecture. The center of considerations in this work is the development and design phase, which sets the boundary conditions on engineering design processes.[22]

1.1.3 Engineering design processes

Engineering design processes are as numerous as the different products, customer requirements and companies. Nevertheless, engineering design research and other academic communities, such as Systems Theory and business management, have conducted intensive analysis of design processes in diverse fields of industry, regardless of company size, nature and complexity of products [BROWNING & RAMASESH 2007]. The goal of the considerations is the provision of models, methods and tools to improve future design and development processes and their planning to enable the synthesis of successful products meeting the time,

[22] The engineering design process underlies certain characteristics exclusively typical for product development, such as the iterative and recursive qualities, which are not transferable or to be generalized for other business processes in production, accounting or any other [WYNN 2007].

cost and quality requirements. Based on significant and acknowledged findings, the following sections show how the design process is structured, how different steps relate to the product architecture and which fundamental characteristics influence product architecture definition and management. Of particular interest are the recent findings in market and organizational structures and their impact on the engineering design process of complex products.

The phase of design and development is situated between the product planning and the phase of production (planning) in the product lifecycle [VDI 2221 1993, p. 8, compare also BROWNING & RAMASESH 2007].[23] Existing engineering design process models range from detailed descriptions of processes in companies, in the form of work packages and workflows [see for example KREIMEYER 2010], which are often typical of the company and the product [LINDEMANN 2009, p. 33-34], to the sequence of fundamental tasks describing the design process in a more rudimentary or generic solution finding process than a business process. Detailed case-specific process models may be analyzed using simulation methods, comparing the competing targets time, cost and quality (or product performance) of processes and resulting products [see e.g. BROWNING 1998, pp. 260 ff., LEVARDY & BROWNING 2009]. In the following sections, the focus of considerations is placed on the generic solution finding process models, to establish a basis for the studies of the product architecture in this work.[24] The analysis of generic process models will shed light on the artifacts and characteristics of product architectures from a process perspective and give the first hints about the requirements for product architecture management in general. In detail, this work is based on the early or conceptual phase, due to the fact that the most successful solutions are more likely to emerge from the definition of the product architecture as a concept of form, function and features [ULRICH & EPPINGER 2003, p. 15], rather than concentrating on technical details in later phases [PAHL et al. 2007, p. 131].

Generic models serve as guidelines for engineering design and have to be adapted to each use case [LINDEMANN 2009, p. 41, PAHL et al. 2007, p. 125, VDI 2221, p. 2]. The intention and benefit of generic process models is comprehensively described by ULRICH & EPPINGER as quality assurance, coordination, planning, management and improvement of processes and their outcomes [ULRICH & EPPINGER 2003, pp. 12-13]. Similarities between different existing models not only show in the intention, but also in the structure of the process models. Generic models do share a sequential proposition of tasks, which are connecting the problem description as an input with the output of the generic process model, the (technical) solution [ULRICH & EPPINGER 2003, p. 12, PAHL et al. 2007, p. 127], the sequence of tasks is in any model intended to detail the product from qualitative to quantitative artifacts describing the product [PAHL et al. 2007, p. 125]. The interpretation of the task's output as an artifact of the

[23] Some authors consider the product planning phase upstream and the downstream activities of testing and production planning/ramp-up part of the design process [see for example the model presented by ULRICH & EPPINGER 2003, p. 14]. In this work, the implications of product planning, testing and production ramp-up are considered as major influences on the design process, but not being part of it.

[24] Detailed process descriptions and models, such as project plans are useful for project management and other business administrative issues [LINDEMANN 2009, p. 16], which will due to that focus not be intensively included into the considerations in this work, which is taking the perspective of engineering design primarily.

product architecture is another similarity that can be observed, along with the necessity to make decisions at the end of each step. To give an example, the Association of German Engineers [Verein Deutscher Ingenieure] (VDI) Guideline 2221 proposes the list of requirements, function structure, principle solution, module structure, preliminary layout, overall layout and product documentation as the outcome of seven tasks of development and design [VDI 2221], while PONN & LINDEMANN focus on the concretization of products with the artifacts of requirements, functions, working principles and a building model as the outcome of the four steps [PONN & LINDEMANN 2008].

The number of tasks, and the detailed description and outcome, differ nevertheless between the numerous existing models, depending on the chosen level of detail, as well as the anticipation of iterative and recursive steps already incorporated into the model. For example, the VDI Guideline 2221 proposes that the task of defining working principles be followed by the tasks of structuring modules and designing these modules, before defining the product as a whole. In contrast, PONN & LINDEMANN see these steps as necessary to be conducted in parallel during all tasks[25] of designing, along with other Design for X aspects, pointing out the relevance of the product architecture throughout all phases of design and avoiding a reduction of the structural considerations to singular tasks [VDI 2221, p. 9, PONN & LINDEMANN 2008, pp. 25-28].

It is critical for the consideration of generic process models to be aware that the process of design is to be conducted iteratively (or cyclically) and recursively. ROYCE already identified the lack of iterative properties as a major flaw in existing procedural models in software development in 1970, pointing out that iterations not only occur between closely related steps, but also involve three or more steps of a proceeding [ROYCE 1987].[26] The recursive character of the design process was manifested by BOEHM for the area of software development in 1988 in the spiral model of development [BOEHM 1988].[27] Today, regardless of which tasks in particular are proposed and on what level of detail, the iterative character of design processes is stated as a fact agreed upon by renowned authors [LINDEMANN 2009, LEVARDY & BROWNING 2009, PONN & LINDEMANN 2008, PAHL et al. 2007, p. 125-126, GAUSEMEIER et al. 2006, p. 29, VDI 2206, CRAWLEY et al. 2004, pp. 4-5, VDI 2221, BOEHM 1988, ROYCE 1987,]. Models such as the Munich Procedural Model in particular are defined as solution finding procedures or processes in product development environments, allowing for a flexible and recursive, as well as iterative, application [LINDEMANN 2009, p. 41]. The basic idea of,

[25] The model of product design proposed by PONN & LINDEMANN is dissected into four major tasks represented by four product models of artifacts, namely the requirements model, functional model, working model and building model [PONN & LINDEMANN 2008, p. 24].

[26] LINDEMANN proposes a flexible networked view on the modeling of processes due to that circumstances, pointing out that the given presumptions are valid in product development as well [LINDEMANN 2009, p. 40ff].

[27] “Recursivity“ is in the context of this work understood as the application of a procedure on different levels of abstraction. As such, the Munich Procedural Model for example is applicable for the planning of the overall design process as well as for singular solution finding processes within the overall process [LINDEMANN 2009, p. 41].

for example the Munich Procedural Model or the VDI Guideline 2221, is the provision of suitable models and methods for the different phases of product concretization [VDI 2221, LINDEMANN 2009]. The use of different models supports the process of analysis and solution finding [KNOBLICH 1997, p. 214-215] although different models rarely interrelate in an iterative and recursive fashion. The transition of models is thus characterized by a loss of part of the available information during transition, analyzed in detail by FUCHS under consideration of general design principles during analytical tasks [FUCHS 2004, p. 76].

Emphasized by such models, these findings allow for the definition of a generic process model closely related to the product architecture depicting the most relevant artifacts and a sequence of tasks suitable for the product architecture considerations in this work. The evaluation of different possibilities of solutions within the different steps, as a core of the discussed models, is as relevant as the provision of appropriate models on which these decisions are based. It is important for the further discussion of the product architecture in this work that recursivity and iteration do have to reflect in the models representing the product architecture's artifacts, to allow for the required recursive and iterative proceeding. The knowledge manifested in these models has to enable taking the correct decisions, taking into consideration the product architecture, as well as the choice of methods. Based on the goals of the project, the correct application has to lead to successful products, the goal of design processes.

1.1.4 Product architecture in industry

As a summary of the previous chapters, the role of product architecture, its implications and influences, as well as the overall challenges in industry are outlined, constituting the basis for a detailed integration of the meaning and relevance of product architectures into a comprehensive product architecture model. The discussion of the situation in industry has pointed to the product architecture as one of the major pivots in product development for the following reasons.

The product architecture poses the link between the tension of the increasing and diversifying customer demands on the one hand, and the increasing technical possibilities on the other. The fulfillment of the evolving market requirements, as described, is the core value of the product. The architecture, through enabling radical innovations [compare HENDERSON & CLARK 1990, p. 12] and functions, incorporating new technology, allowing for variant rich products and containing valuable information, provides the possibility to resolve this conflict.

The key for the product architecture to develop its full potential lies in the considerable upstream shift within the development process, when the influence on the product properties is at its highest.[28] It poses a major challenge in these early phases that the available product

[28] Bullinger et al. consider the upstream shift of activities as one of the major premises to improve development processes (among the use of a project master plan, management of process interfaces, and the feedback from production and field experiences) [BULLINGER et al. 2003, p. 69].

information, though highly relevant and crucial, is not yet fully available and quantifiable. Not all Design for X aspects can be sufficiently addressed at this point.

The importance of the product architecture propagates onto the company, i.e. the organization, processes, knowledge management, decision-making, value networks etc. The dilemma between information availability and the importance of available information in the early phases is thus even more crucial. As a result of these circumstances, the design processes are conducted in a recursive and iterative manner to overcome these difficulties. Although the clear and scientifically sound distinction of wanted and unwanted iterations is not yet conducted, the necessity of iterations to a certain amount is commonly agreed upon. The relevance of recursive procedures is equally acknowledged, due to the increasing amount of information during the process. The information about the design is subject to changes, as well as to an evolution of detail along the design process.

In literature and surveys, the comprehensive situation (as summed up in this chapter and described in detail in the previous chapters, in addition to the numerous interdependencies of the product architecture with its surroundings) is often plainly considered as complexity in product development. The following chapters will not only analyze the nature of the term "complexity" and the subject of complexity itself, but also relate that to the situation of product architectures, to allow for an approach to successfully manage the product architectural issues during the design process with focus on the early phases of design.

1.2 Objectives and problem description

The goal of the presented work is to support the coping with complex product architectures in complex development environments, as described in the introductory chapters. The work should support the overall mission of research in systems architecting, which CRAWLEY et al., among others, describe as "to identify a set of principles, methods, and tools that will help systems architects in the future"; this is valid for both the process of architecting, as well as for the product architecture itself [CRAWLEY et al. 2004, p. 9]. As CRAWLEY et al. state, there are many methods existing that support the analysis and synthesis or other aspects of systems architecting [CRAWLEY et al. 2004, p. 9].

1.2.1 Vision

In support of the management of complex product architectures, it is the vision of the presented work to analyze the existing models, methods and approaches, identify the missing constituents for a comprehensive approach, and provide a framework of methods and approaches, underpinned by a suitable modeling approach, as well as a capable proposition of a procedural model supporting the process of applying the methods and approaches proposed. The fundamental requirements, stemming from the situation of product architectures in industry elaborated in the preceding chapters, are specified in the following chapters, leading to an intensive discussion of the modeling and coping with systems architectures in the following work to refine the requirements for the approach.

1.2.2 Development process and lifecycle requirements

Stemming from the discussion of the product lifecycle, the importance of the early phases became evident, as the influence on the constitutional product properties through the design of the product architecture is highest. Aside from that, the crucial tension between the importance of the early phases and poor information availability has to be considered. The difficulty in regarding the whole lifecycle, especially the evaluation of product properties and the development of sensible variant strategies, are major challenges to be considered as much as possible, even in these early phases.

The iterative and recursive development process poses requirements for the method. The models and methods that are currently known pose a sequence in the process of product architecture definition, but lack integration and transferability for iterations and especially recursive activities. As a result, the research conducted in this work must provide the means to cope with both iterative and recursive applications of models and methods.

1.2.3 Requirements from the stakeholder perspective

Numerous viewpoints regarding product architecture exist, which the systems architect has to harmonize in order to provide a successful product. Relevant Design for X aspects have to be identified and monitored, and conflicts of goals must be solved. Different stakeholders, who evoke the numerous aspects to consider, are for example the customer and different departments within the company, as well as partners within the value network or outside, such as competitors or the legislative body. These circumstances have to be incorporated in a comprehensive approach and its supplementing models and methods.

1.2.4 Requirements on the modeling of systems architectures

The systems architecture not only has to incorporate the requirements stemming from the lifecycle and stakeholder perspective, but also support the innovation process through models, which are of continuous use throughout the innovation process. The requirement for both continuity and comprehensiveness stems from the lifecycle perspective. Requirements stemming from different stakeholders have to be considered according to the multiple levels on which they are fulfilled and evaluated within the systems architecture. Information in general, as a crucial part of decision-making at numerous points in time of the process, has to be incorporated, or at least being related to the approach.

2. Background and classification within the academic domain

Given the objectives of this work and the description of the problem based on the current situation, outlined in the introductory chapter, chapter 2 clarifies the groundwork for a solution approach. The author's background is detailed, providing the perspective under which the presented work was conducted, and enabling the reader to classify the findings and general approach. The scientific aspects include the identification of a research method to define a reasonable approach and thematic classification of the work presented. The classification at this point clarifies which areas of research are affected by the topic and how they constitute the starting point for research on the subject. A brief discussion of research areas will show the potential contributions of the different areas to the solution. Included in this chapter is the logical structure of the thesis as a result of the chosen research approach and relevant research areas.

2.1 Background of the author

The author conducted the presented research during his affiliation at the Institute for Product Development at the Technische Universität München. Numerous research projects, carried out with government funded and in cooperation with industry, were conducted from 2004 to 2010. The author's focal research topics were the analysis, design, and improvement of processes on one hand, and the analysis, synthesis and modularity of technical systems and their architectures on the other. The knowledge about design process provides the foundation for embedding the research on product architectures into a practical context.

The research directions in design processes encompassed different facets, for example the role of people in design processes. Examples of this are the efficient communication and collaboration in virtual product development environments and the knowledge management in design in general, especially in the interfaces to other processes and organizational units. The monitoring of the product maturity in virtual product development was considered from a product perspective. A transdisciplinary research project was initiated, encompassing numerous dynamic and complex interrelations between the multiple different entities involved in the innovation process, considering both the temporal and contextual dependencies. Subsuming these aspects under the notion "Management of cycles in innovation processes", different disciplines, such as engineering, informatics, psychology, and business management, discuss the improvement of innovation processes from different angles. The author's involvement in both the research proposal and project funded by the Deutsche Forschungsgemeinschaft (DFG) provided major insights and contributions to the research presented in this work.

The research on the analysis and synthesis of technical systems included numerous projects in the automotive sector, for example the analysis of noise emissions of mechatronical components and the definition of product property profiles and comparisons. Other branches considered are, for example electro pneumatic brakes for railway applications. Considerations regarding synthesis, with respect to modularity issues, included for example automotive seats, cell phone masts, standardized valve technologies, and energy-efficient drivetrains. The definition of the approach presented in this work includes the experiences and findings of all related projects, though not all aspects can be shown on the basis of the presented case study. The projects in the area of product architecture analysis and synthesis allowed for the application, definition, and evaluation of numerous methods and approaches provided by systems engineering, design research and other research areas. The methods are discussed in this work, as is the practicability in the context of product architectures. In general, the findings of the presented research are based on the numerous aspects evolved in the series of projects conducted.

In addition to the research projects discussed, the author was involved in conducting surveys with industry partners. The survey topics include communication in design, change management, and measures against product piracy. The survey findings are relevant for the presented research; they underline the importance of the transparent modeling of interdependent information, the criticality of changes and their understanding in design, and the iterative and recursive character of the design process, especially among a wide and ambitious field of competitors.

2.2 Research method

Different authors describe the field of engineering design as ambiguous and of fragmented nature [TATE & NORDLUND 2001]. As part of the field of engineering design research, the presented work aims to be classified within this field. Applied research methods often remain unexplained and undiscussed in publications [TATE & NORDLUND 2001], which is why this work aims for a transparent use of research methodology and a sound approach. The following paragraphs will differentiate possible goals of engineering design research, categories according to how the research is conducted, and the relevant subjects or areas of knowledge to position the outcomes. Concluding the section is the discussion of different procedures to conduct research in engineering design, ending with the introduction of the chosen research method for the presented work and its classification.

The **goals** of research in engineering design claim to contribute to either the *improvement of the practice of design* in industry, *to design science* in general (answering scientifically relevant questions and positioning design science in the area of science overall), or to the *improvement of the education* of design [HUBKA & EDER 1996, pp. 74-75].

The main **subjects of consideration** in engineering design research are the *product* and the *design process* [HUBKA & EDER 1996, p. 82, BLESSING & CHAKRABARTI 2009, p. 5], while the areas of knowledge in design theory are the design process, design object, designers, specific field knowledge and resources (e.g. time, money) [TATE & NORDLUND 2001].

HUBKA & EDER furthermore provide three **categories** of how engineering design is conducted: *research into design*, i.e. observations for understanding design (“experimental research” according to BLESSING & CHAKRABARTI); *research for design*, including the creation of models, methods and tools (“intellectual” according to BLESSING & CHAKRABARTI); and *research through design*, i.e. self-observation (described by BLESSING &

CHAKRABARTI as “experiential”) [BLESSING & CHAKRABARTI 2009, p. 3, HUBKA & EDER 1996, p. 38].

Given the different foci, approaches, and subjects of engineering design research, the demand for generic procedures to conduct and assess research results has arisen. Different authors provide adequate procedures, many of which are based on the differentiation and coupling of descriptive and prescriptive methods. The development of methods should, in general, include the three steps of data gathering, theory generation (generate hypotheses, relate to theories), and theory validation (deduce consequences, test) [TATE & NORDLUND 2001]. WOOD & GREER, as well as BLESSING & CHAKRABARTI, largely base their approaches on the differentiation between prescriptive and descriptive phases, accompanied by the identification of success criteria, impact chains, and the testing of the generated outcome against these consdierations [WOOD & GREER 2001, BLESSING & CHAKRABARTI 2009]. A cutout of the Design Research Model of WOOD & GREER is depicted in the following figure, showing the relevant aspects for the development of methods in engineering design research for the presented research [WOOD & GREER 2001, p. 177].

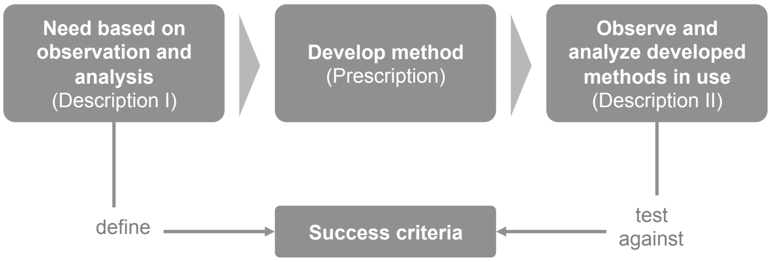


Figure 2-1 Cutout of the Design Research Model according to WOOD & GREER [WOOD & GREER 2001, p. 177]

While some of the authors mentioned provide generic, yet comprehensive, procedures, BLESSING & CHAKRABARTI provide a framework for engineering design research; this supports distinctive steps with appropriate methods, points out different ways to achieve valuable results, and allows for the classification of research outcomes [BLESSING & CHAKRABARTI 2009]. The following paragraphs include the classification of the presented work within the above framework for the engineering design research methodology, while other chapters incorporate the concrete overall objectives (chapter 1.2) and refined goals (chapter 5.8).

The **goal of the presented research** is the improvement of design, in particular coping with product architectures. As a secondary goal, the work aims to significantly contribute to design science by giving answers to scientifically relevant questions regarding how to cope with evolving architecture knowledge along iterative and recursive design processes.

The **subject of consideration** of this work is mainly the product architecture itself, in reference to its entities and evolving character. The underlying design process serves as a secondary subject, since the characteristics of design processes provide the main challenge and demand for the presented methods and models.

Since methods and models are the core outcome of the presented work, it is the **research for design** that best categorizes the conducted research. Experiential aspects (compare chapter 2.1) additionally contributed to the procedure as well as the outcome and methods.

Based on different approaches and theories, the presented work is structured according to the findings of previously mentioned authors. First, **descriptive study I** describes the role of the product architecture as discussed in chapter 1, ending with the definition of the problem and the according requirements for a solution. Descriptive study I closes with a discussion of relevant fields of research in chapter 3, laying the groundwork for the first prescriptive study, as well as the following descriptive studies. Descriptive study I clarifies why dealing with product architectures is relevant, both from an industrial as well as scientific point of view, based on literature research.

Prescriptive study I establishes a framework for systems architecting, reaching conclusions based on the findings of descriptive study I. The framework identifies a modeling method, as well as the entities of the product architecture and classes of entities relevant for the management of product architectures (chapter 4).

The established framework is tested against the existing methods discussed in **descriptive study II**. Descriptive study II discusses not only the role of the product architecture, but also existing methods and methodologies to cope with different entities of the architecture within the distinct phases of product development (chapter 5). Based on an extensive literature review, descriptive study II allows for the detailed definition of the demand for action from scientific as well as industrial perspective despite existing activities (chapter 5.8).

To meet these demands, **prescriptive study II** introduces the missing elements for a comprehensive approach (chapter 6), and combines these with the methods and procedures identified in chapter 5. The results combine the framework (chapter 4) and methods for systems architecting (chapters 5 and 6) to allow for a comprehensive solution approach (chapter 7).

The approach is then tested against the objectives through the application within a project-based verification. This **descriptive study III** represents a use case-based example of application (chapter 8).

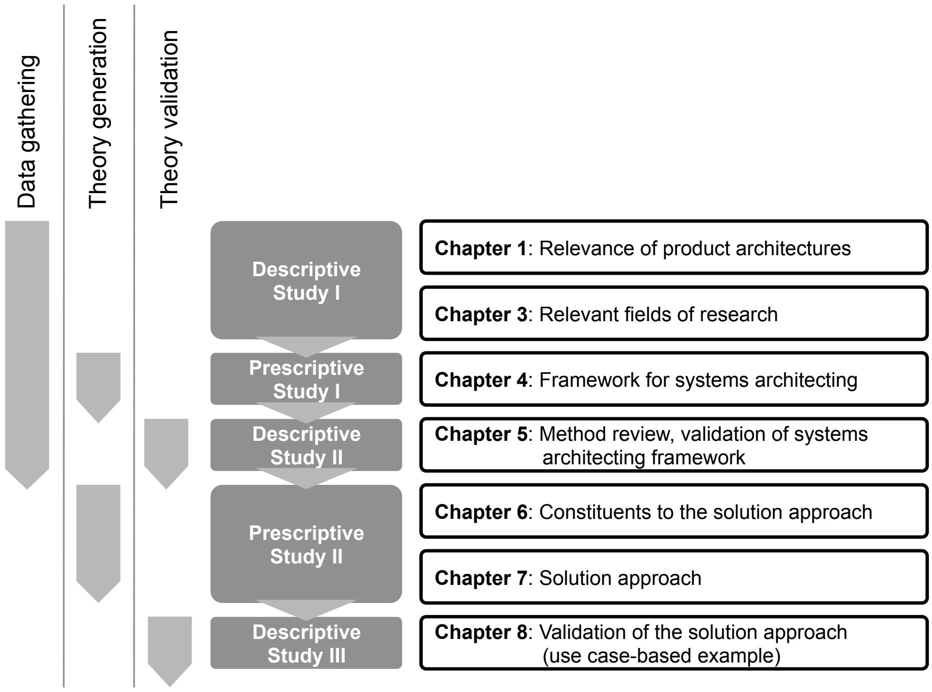

Data gathering
Theory generation
Theory validation
Descriptive Study I
Prescriptive Study I
Descriptive Study II
Prescriptive Study II
Descriptive Study III
Chapter 1: Relevance of product architectures
Chapter 3: Relevant fields of research
Chapter 4: Framework for systems architecting
Chapter 5: Method review, validation of systems architecting framework
Chapter 6: Constituents to the solution approach
Chapter 7: Solution approach
Chapter 8: Validation of the solution approach (use case-based example)

Figure 2-2 Research approach of the presented thesis

In addition to the research method, the presented work is intended to fulfill the "demands of science", as expressed by HUBKA & EDER [HUBKA & EDER 1996, p. 75], which rightly call for research to fulfill the following qualities [HUBKA & EDER 1996, p. 38]:

- Purposeful (identification of a problem worth researching)
- Inquisitive (seeking to acquire new knowledge or new relationships among knowledge elements)
- Informed (conducted from an awareness of previous research)
- Methodical (planned and carried out in an efficient and disciplined manner)
- Communicable (testable and accessible results)

2.3 Classification within the academic domain

Based on the defined scope of the thesis in chapter 1, this research work is placed within fields of research that cope with the management of complexity, engineering design and innovation processes. The areas of research, discussed in the following chapters, are partly overlapping and use one another's approaches and theories. As such, the presented approach aims for the identification of contributions from the different areas and the application within the field of engineering design research.

To cope with complexity within organizations, processes, and products, it is clear that the areas of systems theory, systems engineering, and networks science and Graph Theory should be more more closely examined. While **systems theory** provides the groundwork for the understanding of systems, **network science and Graph Theory** apply mathematical models and methods to large system structures; the presented work was motivated by this area of research. **Systems engineering** as part of Systems Theory philosophy provides a more concrete means for the area of engineering design research, based on the system understanding of systems theory.

Operations research primarily provides a means for decision-making in complex situations, and through that contributes to the management of product architectures, as the process of engineering design is characterized by a large number of decisions to be made throughout the process.

Engineering design research as a broad field provides many distinct measures for product development and development processes, ranging from social aspects, concrete methods to improve the outcome of the respective steps of the design process, process optimization etc. Since the field of product architecture management is a rather recent development, existing approaches have to be discussed in that context.

The above fields of research are introduced and discussed in detail in chapter 3.2. The methods and approaches from the fields are evaluated and discussed in chapter 5. While many fields provide valuable input for the presented work, it to is the fields of engineering design research and systems engineering that the results of this work intend to contribute, based on the identified shortcomings within the discussion of the state of the art.

2.4 Structure of the thesis

Following the introduction of the topic in section 1 and the outline of the research in the previous sections, the structure of the thesis is defined as follows:

Section 3 discusses the notion and character of complexity, as well as the areas of research in question for coping with complex product architectures. As an outcome, the core understanding of complexity and coping with complex systems is defined, and provides the groundwork for the definition of the presented work, as well as for the discussion of the state of the art.

The systems thinking perspective is narrowed down to the scope of product architectures in **section 4**, appropriate modeling techniques, and a classification of architectures. Based on relevant literature, the defined framework for this approach allows for the detailed discussion of the state of the art in the following section**.**

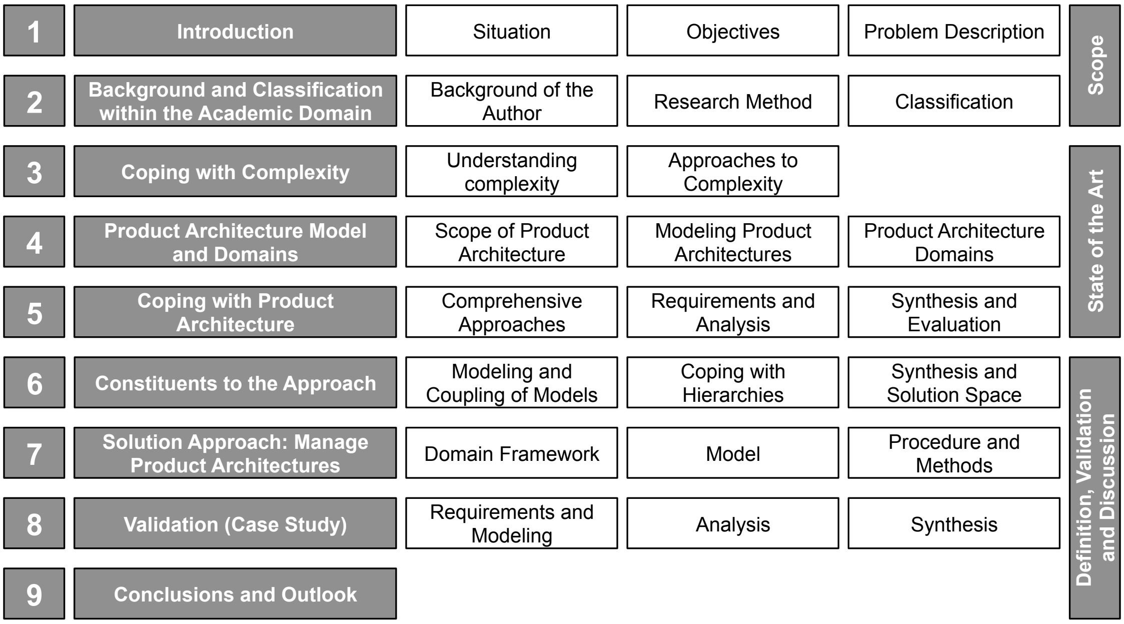
Section 5 discusses in detail the available procedures and methods intended to cope with the product architecture; as a result, suitable methods and gaps in actual research are identified. As a conclusion to this section, the requirements of the approach are defined, structured accordingly to the state of the art.

To close the identified research gaps in order to fulfill the identified requirements, **section 6** proposes constituents to the approach, which are, as yet, uncovered by the state of the art. These constituents present the completion of methods, in combination with those from the state of the art, making a comprehensive approach on product architecture management feasible.

The comprehensive approach is presented in **section 7**, identifying appropriate steps for a product architecture management approach, a comprehensive model, and feasible methods for each step.

To validate the approach, **section 8** delivers an example from the automotive industry, providing a sample application and possible outcomes of different parts of the approach. Concluding this section, the results are critically discussed and further requirements and research identified.

Section 9 concludes the thesis, summing up the procedure and outcome of the research approach, and identifying future activities for the field of product architecture management.



1	Introduction	Situation	Objectives	Problem Description	Scope
2	Background and Classification within the Academic Domain	Background of the Author	Research Method	Classification	
3	Coping with Complexity	Understanding complexity	Approaches to Complexity		State of the Art
4	Product Architecture Model and Domains	Scope of Product Architecture	Modeling Product Architectures	Product Architecture Domains	
5	Coping with Product Architecture	Comprehensive Approaches	Requirements and Analysis	Synthesis and Evaluation	
6	Constituents to the Approach	Modeling and Coupling of Models	Coping with Hierarchies	Synthesis and Solution Space	Definition, Validation and Discussion
7	Solution Approach: Manage Product Architectures	Domain Framework	Model	Procedure and Methods	
8	Validation (Case Study)	Requirements and Modeling	Analysis	Synthesis	
9	Conclusions and Outlook				

Figure 2-3 Structure of the thesis

3. Understanding and coping with complexity

As outlined in chapter 1, complexity poses one of the major challenges in the development of demanding technical products. Complexity does not only emerge from within the product architecture itself, but occurs within or originates from different areas, such as markets, processes etc. To define appropriate measures when dealing with complexity, the following sections will present the perceptions of complexity from different points of view. The discussion will clarify the influencing factors and how they contribute to an extensive understanding of complexity. In the subsequent chapters, the focus of different disciplines will be introduced, allowing for the comprehensive discussion of appropriate measures. Systems Theory provides a solid basis, and Operations Research and Systems Engineering are introduced as followers of that school of thought. Business management and engineering design research, as well as design theory, aim for the practical application of different approaches to manage complexity and provide subject matter of the concluding chapters. This chapter introduces the scientific perspective of complexity, and methods that deal with complexity from a generic perspective or can be generalized.

3.1 Understanding complexity

Complexity occurs in almost all industries. Not only is it spoken of in publications and everyday life, but tends to dominate the human perception of its environment in the ongoing century [VESTER 2001, p. 22]. Therefore, it is reasonable to briefly discuss the term "complexity" in this section to gain a common understanding of its influencing factors, properties and diverse appearance; in particular the fact that definitions of complexity slightly vary from one another in different publications. This results in a lack of a standardized notion of the term [WEBER 2005b], leading to the conclusion that a discussion of the term is relevant for the context of this work to make explicit the perceptions on which the following considerations are based. In this context, it is not the goal to provide a general definition valid for all researchers and practitioners, but to underline the prevailing understanding of this work and to base this perception on existing schools of thought.

In the beginning of this chapter, the system aspect of "complex system" is analyzed, and its definitions and characteristics investigated. Based on these considerations, the subsequent section derives the influencing values whose characteristics divide complex systems from non-complex systems. The resulting perception of complex systems is outlined in the last section of this chapter, allowing for a comprehensive discussion of appropriate measures in the field of product architecture management in the following chapters. The main reason for the abstract approach to product architectures is the possibility of identifying and applying measures for managing complexity from different areas thus extending the range of applicable principles, methods and methodologies. While Systems Theory evolved due to the identification of identical phenomena in different areas [PULM 2004, p. 21], the presented work aims to benefit from that by abstracting product architectures to discuss the applicability of the different generalized principles.

3.1.1 Anatomy of a system

It is clear that the foundation of the definition of complexity should be based on a more abstract definition of the object of consideration. Systems Theory therefore provides the foundation to describe a more abstract system than the common models of systems. Appearance and content of common models usually depend on the context in which the user of the model requires information. A model in general is a reproduction of reality suitable for this context [FUCHS 2004, p. 18]. The goal of the model as such is to provide the necessary information in a way the user is able to cope with [DAENZER 1979, p. 13, PAHL et al. 2007, p. 28-29].[29] To allow for a more universal description, a system has to be described in an abstract manner, making the description applicable for different types of systems. Objects of consideration could be, for example, different product architectures, but also systems of other areas, such as development processes, organizational structures or systems not related to engineering design at all, such as social or biological systems. In general, any object which can be distinguished from its environment is considered a system [ROPOHL 1975, p. 25]. PULM identifies approaches in or closely related to Systems Theory in the areas of biology, sociology, psychology, engineering and business management, mathematics, physics and information technologies, as well as in philosophy and linguistics [PULM 2004, p. 23].

The first qualifying prerequisite for system definition is the possible differentiation between the system and its environment [ROPOHL 1975, p. 25, SIMON 1996, p.11, DAENZER & HUBER 1999, p. 6], each separated from the other by the system boundary [PAHL et al. 2007, p. 27].[30] Reasons for drawing a line between the system under consideration and its environment can be numerous, especially in a technical context. Considering, for example, a technical product, the environment might consist of another product, which the product under consideration is part of. The environment could also consist of the situation in which the product user finds himself, which again might be the traffic on the streets, a railroad system or the user's house, depending on the product under consideration. Generally speaking, the "inner system" is an arrangement of elements to obtain the system's goals, for which the "outer environment" defines the prevailing conditions under which the system's goals must be achieved [SIMON 1996, p.11]. For the designer involved, the differentiation between system and environment helps to keep the system under consideration manageable [STEINMEIER 1999, p. 15] by focusing on the core of the problem, viewing the environment as boundary conditions or defined interfaces.

Systems can be decomposed into subsystems and elements [ROPOHL 1975, p. 28, DAENZER & HUBER 1999, p. 7, PAHL et al. 2007, p. 27]. A system itself, on the other hand, is part of a superordinate system and, in that role, is a subsystem from the point of view of the superordinate system [ROPOHL 1975, p. 30]. Complex systems turn out to be decomposable in most cases, enabling the viewer or operator of the system to grasp the system and its

[29] The bill of materials and a CAD model, for example, are both models of a product architecture, yet address different users (or stakeholders) and therefore appear in different representations and contain different, although overlapping, information.

[30] SIMON differentiates between "inner system" and "outer environment" [SIMON 1996, pp. 9 ff.].

complexity [SIMON 1996, p. 207]. The decomposition of a system usually takes place within the same domain or category, such as physical parts of a product. The decomposition of physical parts, for example, might consist of the decomposition of the assembly into subassemblies and parts. Nevertheless, the superordinate system of the product might be of a different, although still physical, type. A frequently used example is the automobile, which can be decomposed into its assemblies and parts, but on the other hand is part of the traffic as a superordinate system. The definition of a system's decomposition has to be strictly separated from the concretization of, for example, functions to working principles to physical parts, where again the functions could be technical or user-oriented and so on. Like the system under consideration, the superordinate system can be viewed from different angles or perspectives; traffic as a superordinate system could be viewed from a functional perspective, as can the automobile. The concretization of a system discussed in the context of the engineering design processes takes a process-related viewpoint, from which the system is pictured in different models, suiting the purpose at the given phase of the process.

System elements and subsystems are interrelated with one another through relations [DAENZER 1979, p. 11, DAENZER & HUBER 1999, p. 5], as the system itself is equally connected to the superordinate system and to the system's environment. Relations across the system boundary are usually considered to be both inputs and outputs of the system [PAHL et al. 2007, p. 27].[31] The types of possible relations are almost immeasurable, as are the imaginable types of elements. In the context of product architecture management, a detailed overview on relevant connection types will be given in the later chapters of this work. The differentiation of decomposition and different perspectives of a system are even more crucial when taking the relations into account. While the decomposition inherits hierarchical relations between system and subsystems, the different perspectives of the system are also interrelated. The physical parts of a product are interrelated with the functional perspective, for example. When modeling a system with such different and diverse perspectives, it is common to establish different models, for example a function structure and a physical layout, to be able to cope with the structure and its characteristics. Different tasks require different models, as was already pointed out, and, as such, a system can hardly be grasped with all of its perspectives and inherent structure(s) simultaneously.

The interrelations of a system are highly relevant for the system structure as, according to DAENZER, the structure of a system results from the formation of elements and relations due to their interrelations, through which the system retains its integrity [CHECKLAND 1993, p. 121, DAENZER 1979, p. 12, DAENZER & HUBER 1999, p. 6]; i.e. given the system's elements and their interrelations, the resulting formation or alignment can be grasped as an abstract attribute of the system, which can be analyzed thoroughly through computerized measures of Graph Theory [compare MAURER 2007, p. 31-32]. The outstanding role of the structure as a system attribute is underlined by DAENZER, stating that situations and systems can only be understood by human perception if they can be assessed structurally [DAENZER 1979, p. 12].

[31] It should be emphasized that the directed *relations* between elements inside and outside the system boundary are considered as input and output, not the *elements* outside the system.

A system possesses further attributes,[32] which can be allocated to the system itself (e.g. cost, weight) [ROPOHL 1975, p. 26-27]. System attributes can be decomposed according to the system, although the interrelations of the system as a whole may differ from the interrelations existing in the decomposition of the system. For example, the acceleration of an automobile results from a concurrence of many different subsystem attributes, such as engine power, transmission, gear ratio etc. ROPOHL defines the consideration of attributes as inputs and outputs of the system as another perception of attributes, describing the function of a system by means of characterizing the difference between input and output of the system [ROPOHL 1975, p. 26-27, EHRLENSPIEL 2009, p. 23]. Although the notion of function in this context is similar to the functional model of other authors, the given definition of function as a generalized attribute of an abstract system is not necessarily relevant for the considerations of this work and rather confusing considering the existing functional structures in engineering design research. BAUMBERGER sums up the perception of ROPOHL as the hierarchical (decomposition), networked (structure) and functional (behavior) view of a system [BAUMBERGER 2007, p. 69].

While the presented functional view of behavior still presents the behavior of a system based on the static interrelations of the system elements, the system dynamics are perceived from a different, structural, point of view. DAENZER characterizes system dynamics as the differentiation between three types of causes for dynamics: namely, the type and intensity of interrelations (between elements of the system as well as between the system and its environment), changes of attributes of elements and changes of the structure i.e. the formation of elements [DAENZER 1979, p. 21]. Changes in the structure include the emergence or dissolving of elements or interrelations. MAURER comes to the conclusion that the dynamic of a system stems from changes occurring in the system in general [MAURER 2007, p. 31].

In the preceding sections, the discussion focused on open systems, i.e. systems that interact with their environment. Closed or isolated systems, on the other hand, have interrelations exclusively within their system boundary and are considered to be a highly simplified way of perceiving a system. Closed systems rarely reflect reality and are thus not part of the conclusion of the systems notion. The contrary point of view, a consideration of exclusively the system environment, is another possible perspective, regarding the system itself as black box [DAENZER 1979, p. 20, STEINMEIER 1999, p. 15]. This perception supports the understanding of a system’s basic function and interaction with the environment, disregarding possible complex interrelations within the system, and not considering them for later solution finding processes [LINDEMANN 2009, p. 108]. The notion of closed systems is not an integral part of the presented definition.

Based on the presented discussion, a number of general conclusions can be made. A system in general comprises of elements interrelated with one other and is connected to an environment through inputs and outputs dissected by the system boundary, while the system itself is part of a superordinate system. The system, as well as its elements, inherits attributes. When working with systems, four distinct views can be differentiated, which are best considered separately.

[32] LINDEMANN defines *attribute* as the combination of a certain *property* (e.g. cost or structure) and a determined *specification* (e.g. 100 € or networked) [LINDEMANN 2009, p. 146].

A system's structure is given by the formation of its elements *(structural view)*. A system can be decomposed into smaller parts *(hierarchical view)*. Input and output of a system describe its overall behavior *(performance view)*.[33] System dynamics are characterized by changes within the system *(dynamic view)*.

3.1.2 Influences on complexity

In the context of this work, the influence values on complexity are considered to be appointed system properties that characterize a system as complex. A distinction has to be made between the influencing values and the origin or causes of complexity, which were discussed in the introductory chapter. The definition of complexity or complex system has to be closely related to the definition of the term system. To bridge the gap between the two notions, the influencing values discussed drive the qualities of a system, so that it is perceived as a complex system. The influencing values are closely related to the definition of a system and its origin. Elements and interrelations are derived from the system definition, and pose the structure and thus the nucleus of a system, its attributes, behavior and dynamics. In accordance with this definition, the consideration of complexity follows the same train of thought.

It is clear that the number of elements and interrelations is considered to be characteristic for complex systems. While the number of elements is considered a necessary characteristic of a complex system, the accompanying influence value is in different disciplines considered to be the variety both of elements and interrelations [see e.g. VON BERTALANFFY 1976, p. 54, SIMON 1996, pp. 183 f., STEINMEIER 1999, p. 17, MAIER & RECHTIN 2000, p. 6, CRAWLEY et al. 2004, p. 14, WEBER 2005B, DANILOVIC 2006, EHRLENSPIEL 2009, p. 36, LINDEMANN 2009, p. 9]. A large number of elements interrelated with one other through many interrelations is far more likely to be considered a complex system if the elements and interrelations differ greatly from one another. The resulting variety leads to multiple perspectives of the system, which are all connected and thus influence each other. Examples of this are the complex interactions of product parts from different disciplines in mechatronic systems [e.g. FELGEN 2007, pp. 42-47, VDI 2206, p. 14], or the interaction of tasks, tools and people [e.g. KREIMEYER 2010] in engineering processes etc. In addition, CRAWLEY ET AL. point out that eventually hidden or unrecognized interdependencies contribute to complexity due to their barely tangible nature [CRAWLEY et al. 2004, p. 14]. MAURER underlines the existence of such interrelations (or elements) by emphasizing the importance of information acquisition in the process of analyzing complex systems [MAURER 2007, p. 94ff].

The given system structure based on elements and relations evokes the behavior of a system, as was stated in the system definition. The behavior is considered an influencing value of complexity; the desired behavior of a system interferes with undesired behavior due to required or hidden interrelations, which provoke unwanted side effects. Systems architects take the undesired behavior into account and accept its existence, which is described as the

[33] The term *performance view* is chosen to avoid misunderstandings with the term *function* as in *functional models* etc.

curse of complexity (in contrast to the value of complexity due to the desired behavior) by CRAWLEY ET AL. The system behavior is not the sum of behaviors presented by the selection of subsets; it is only achieved by the interrelation of all subsets. Designers accept undesired behavior to achieve the desired behavior [CRAWLEY et al. 2004, p. 2]. Attributes of subsystems and elements might further increase the conflicts of objectives observed in system behavior [LINDEMANN 2009, p. 89ff].

The occurring changes in systems increase complexity even further, contributing in an important manner to overall system complexity [CRAWLEY et al. 2004, p. 14]. Changes of interrelations and elements, noticed or unnoticed, cause changes in both the system structure and behavior, according to the system definition. AHLEMEYER & KÖNIGSWIESER describe changes or uncertainty as the selectivity of the system, i.e. not all possible interrelations exist at the same time [AHLEMEYER & KÖNIGSWIESER 1998, p. 26-27]. Changes occurring over time may be due to the behavior and structure of the system or generated by the systems architect or user, and, together with hidden interrelations, cause the main uncertainties in handling systems in most cases.

3.1.3 Concept of complexity

A working definition of complexity for the field of product architectures is derived from the system definition and the influencing values. Influencing values can be summarized as number, variety, uncertainty and undesired behavior. In accordance with the different perspectives of systems, different types of complexity can be derived.

Structural complexity is caused by the number and variety of system elements and interrelations, leading to different possible perspectives of the system and increasing the chance of unintentionally ignoring certain types of elements or interrelations, due to an perspective of the system not taken. *Behavioral complexity* results from the interplay of subsets of a system, which causes the occurrence of undesired and/or unexpected behavior, due to hidden or undesired, but necessary, interrelations. *Dynamic complexity* occurs due to the number and variety of known or unknown, desired or undesired changes to the system. Dynamic complexity is intensively affected by the uncertainty or selectivity of a system, causing unforeseeable changes and thus unexpected dynamics.[34]

While the definitions above are, in principle, objective measures of complexity[35], CRAWLEY et al., as well as EHRLENSPIEL, introduce *interface complexity* as a subjective matter of complexity [CRAWLEY et al. 2004, p. 14, EHRLENSPIEL 2009, p. 36]. Interface complexity is

[34] The consequential progression of the different views of the system definition would imply the existence of *hierarchical complexity* as well. As hierarchical decomposition as such is one of the most common principles to reduce complexity [see e.g. AHLEMEYER & KÖNIGSWIESER 1998, p. 22], it does present a different view of the system, but does not add further aspects to the concept of system complexity beyond that.

[35] CRAWLEY ET AL. state that the mentioned types of complexity are objective measures because they are theoretically measurable. Constraints for the measurability are the availability of concrete measures and a common agreement on those measures [CRAWLEY et al. 2004, p. 14].

experienced by people interacting with the system, e.g. people involved in the downstream activities of a technical system, such as users, operators or assemblers [CRAWLEY et al. 2004, p. 14]. EHRLENSPIEL describes this situation as complicated for the user, but not necessarily complex [EHRLENSPIEL 2009, p. 36].

3.2 Approaches to complexity

The numerous concepts of complexity evoked distinct approaches to deal with the subject in different disciplines or schools of thought. While each discipline identified complexity in its own field and developed adequate measures to deal with it in its respective environments, theories and approaches developed over time, approaching complexity from a more general perspective. The groundwork and basic principles to enable any common understanding stem from Systems Theory or cybernetics.[36] Systems Theory in general seeks to combine known and familiar patterns in systems from different disciplines, such as biology, sociology, psychology, engineering, business management, philosophy, linguistics etc. [PULM 2004, p. 21]. On the other hand, the individual disciplines withdraw dedicated elements, i.e. models, mathematical operations or general trains of thought for problem solving etc., from Systems Theory and incorporate these elements into their overall proceeding and methodology. It is a ceaseless undertaking to precisely distinguish different fields of research in Systems Theory and define which came first. Although systems thinking and Systems Theory emerged to establish a system understanding independent of certain disciplines and application areas, the respective disciplines nevertheless tend to differ from one another, be it by developing a discipline-specific application and extension of general Systems Theory or the relevant context alone in which the methods and models are presented [ROPOHL 1975, pp. 22-24].[37] CHECKLAND gives an extensive historical overview on systems thinking and Systems Theory, citing numerous groundbreaking publications from different disciplines in the past century and shedding light on the detailed history of Systems Theory [CHECKLAND 1993, pp. 59-98]. BERTALANFFY, being one of the first researchers of the field, points out the early history of Systems Theory [VON BERTALANFFY 1976, p. 10-17]. Systems Theory encompasses the three aspects of systems science, systems technology and systems philosophy [PULM 2004, p. 21], of which general systems science[38] and system technology[39] will be considered intensively; although the philosophical perception and contribution is valuable, it provides no immediate addition to the topic. The following disciplines will be outlined in the subsequent chapters and their perspectives on Systems Theory discussed. Operations research, originating in the late 1930s in the military sector, focuses on decision-making during the planning of activities or

[36] SIMON gives an even broader historical overview, including Holism and Reductionism, Catastrophe Theory etc. [SIMON 1996, pp. 169 ff.].

[37] ROPOHL for example is positioning cybernetics in the center of his considerations, defining Systems Theory, systems engineering, operations research, informatics, organizational science and scenario planning as related areas among “others” [ROPOHL 1975, p23].

[38] The notion systems science is here used synonymously with the notions Systems Theory or cybernetics.

[39] System technology is used as an equivalent term to systems engineering.

processes on the basis of mathematic models derived from descriptive models of the problem [see e.g. DOMSCHKE & DREXL 2002, ZIMMERMANN & STACHE 2001]. Network Science, originating in the 1990s, concentrates on structural complexity from the perspective of networked systems, strongly based on the definition of systems as a construct of elements and their interrelations. It is closely related to Graph Theory, providing the mathematical groundwork for complex problems [DIESTEL 2006]. The focus of Network Science lies in large networks with numerous elements and interrelations to identify patterns on a statistical basis. Graph Theory or Network Science are used as means in Operations Research, of which they were originally considered a subset [DOMSCHKE & DREXL 2002, p. 8]. Systems engineering treats complexity application oriented from an engineering perspective with a strong link to business administration and project management, providing concrete methods and procedures for successfully developing and dealing with complex systems in business [see e.g. KOSSIAKOFF & SWEET 2003]. Software engineering, business management and engineering design research provide models, methods and tools relevant for the respective area, derived from or combined under the notion of Systems Engineering.

The approaches cannot be discussed exhaustively in this work; the following chapters will give an overview of each topic based on relevant literature, allowing for the discussion and contribution of each area to the overall goal of the management of product architectures. The overview was conducted to develop an understanding of the different schools of thought to structure the existing approaches. Therefore, the following chapters will proceed from the rather generic to the more specialized approaches, introducing the different approaches briefly outlining their history, aims, subjects of consideration, methods, models and languages, and the relevance for the research introduced in this work. The most promising models, methods and tools, in general the relevant subsets of an approach, will be discussed in detail in the chapter 5, highlighting existing applications and methods stemming from the discussed schools of thought.

3.2.1 Systems Theory

Since its origin in the 1920s, general Systems Theory pursues theoretical model building between theoretical and generic mathematical models and the specialized models of the separate disciplines. General Systems Theory is one of the few “global theories” that is still present in science. Although the more specialized forms of it, such as e.g. systems engineering, are far more popular in research and practice today, it is the basic understanding of systems, and the sensitization for the topic stemming from general Systems Theory, that is still cited and representing the basic understanding of systems to present [BOULDING 1956]. A historical overview on early Systems Theory is provided for example by BERTALANFFY, who, cited by many, was also one of the first researchers of the field [VON BERTALANFFY 1976, p. 10-17], while Checkland provides a broader overview on the history of Systems Theory and its placement in the history of science overall [CHECKLAND 1993, pp. 59-98]. Systems Theory encompasses the three aspects of systems science, systems technology and systems philosophy [PULM 2004, p. 21]. The following discussion will concentrate on systems science with remarks to systems philosophy where appropriate, while Systems Technology will be discussed as Systems Engineering.

The following aims of Systems Theory, based on BERTALANFFY, BOULDING and CHECKLAND [VON BERTALANFFY 1976, p. 38, BOULDING 1956, CHECKLAND 1993, p. 93] are loosely ordered according to the estimated probability of achieving them: centering the integration of various sciences to actively seek for isomorphic[40] concepts, laws or models in the respective fields; investigating of useful transfers from one field to another; enabling the development of adequate and exact models and theories for the disciplines which lack them, especially nonphysical disciplines; revealing gaps in existing theoretical models of individual disciplines by providing a system of systems in science; eliminating parallel efforts in different fields; and improving integration between specialists in science, practice and education, thus promoting the unity of science. It is important to note that the "unity of science" does not refer to a general encompassing theory making all specialized disciplines obsolete, but rather to an improved integration of disciplines by common theories and models[41] [BOULDING 1956]. As generality commonly causes the loss of content of an approach, method or model, it is not the goal of general Systems Theory to substitute all other disciplines, but to provide consolidated findings of a more generic applicability [BOULDING 1956]. Neither is it the goal of Systems Theory to reduce all sciences to the laws of physics, as was the perception of a unification of science before Systems Theory [VON BERTALANFFY 1976, p. 48]. The results are meant to be applicable to different disciplines and provide insights based on the generic theories on systems [BOULDING 1956]. STEINMEIER rightly states that, from the Systems Engineering perspective, the system-oriented approach in practical application has to supply solutions to specific problems, a notion which is valid in other disciplines as well. This underlines the importance of the choice of perspectives to be taken, dependent on the system situation- [STEINMEIER 1999, p. 14], as was determined within the discussion of the understanding of system and complexity. The foci of research and practice, namely the identification of generic findings and the specific application, nevertheless do not represent a contradiction. The gap highlight the need for a more specific definition of approaches to sufficiently apply Systems Theory in practice, culminating in research areas, such as Systems Engineering, and leading to the implementation of the fundamental ideas of Systems Theory into other research areas, such as engineering design research and business management.

Following the definition and goals, the subject under consideration in the conducted research in Systems Theory, namely "the system" itself, cannot be limited to a certain number or particularly defined systems. Instead, numerous classifications or typologies of systems were developed by different authors, pointing out the diversity of the systems considered in Systems Theory and allowing for an improved differentiation of applicability of particular

[40] *Isomorphy* has varied definitions in different disciplines. In Systems Theory, especially from the perspective of systems engineering taken in this work, it can be defined as equivalence and validity of a model or theory in different disciplines accompanied by the possible qualitative or quantitative reproduction of a part of each discipline's reality.

[41] The developing and sharing of models and theories in cooperation between different disciplines is typically referred to as *transdisciplinary*, as opposed to *interdisciplinary*, i.e. the mere "working together" of disciplines, an integration on a level above the working level of the disciplines, without sustainable long-term impact on the disciplines [for an extensive discussion of the terms, see for example BRAND et al. 2004 or LASZLO 1995].

models and theories in different types of systems. A common differentiation is the distinction of hard and soft systems. While hard systems are precisely definable and represented mostly by systems in engineering, such as design problems, soft systems constitute the fuzzy problem situations set in e.g. social systems [CHECKLAND 1993, p. 189-191]. BERTALANFFY divides hard systems into closed and open systems, of which closed systems are, for example thermodynamical processes, while open systems interact with their environment, allowing the exchange of inflow and outflow across their system boundaries, if boundaries can be clearly defined at all [VON BERTALANFFY 1976, p. 39]. PULM considers the evolution of Systems Theory as a shift from closed to open and finally to soft systems, i.e. from general Systems Theory to first and second order cybernetics [PULM 2004, p. 23]. It is important to recognize this "paradigm shift" to understand that increasingly complex systems can be influenced, rather than controlled, due to their openness, instability, dynamic, indetermination: in summary, their increasing complexity [PULM 2004, p. 43]. Clearly, even the fundamental distinction between hard and soft systems has blurry boundaries, as an engineering problem can be tackled in the early requirements- and customer-interrelation phases through soft system methodology, giving credit to the social and human aspect of a product's purpose. Fuzzy methods are also of increasing use in engineering [see e.g. BONJOUR et al. 2009, WERTHNER 1994 pp. 99ff.]. In particular, Systems Engineering and the early phases of design profit from fuzzy methods in situations where problems cannot be stated clearly [CHECKLAND 1993, p. 191]. On the other hand, concepts of Systems Engineering methods, such as the definition of domains and their interrelations as proposed by MAURER, prove to be useful to enhance for example the general approaches of system dynamics [MAURER 2007].

Widening the scope, MALIK provides a distinction with similar results, differentiating systems on the basis of their origin into technical, social and natural systems. Following this argumentation, technical (hard) systems evolve due to both human intention and action, while complex social (soft) systems develop without human intention, but require human action. As a third category, MALIK introduces natural systems, which evolved without human intention or human action, such as the planetary system. To complete the picture, according to MALIK no system exists without human action but stemming from human intention [MALIK 2008, p. 219]. CHECKLAND, besides his general differentiation of soft and hard systems, supports MALIK's perception of system classes, adding only human activity systems, i.e. sets of human activities viewed conjointly as a system, to the list of natural, designed and social systems. Human activity systems, according to CHECKLAND, are different from natural systems due to human self-consciousness and human choice, resulting in different possible occurrences of the system as opposed to natural systems. The difference from social systems stems from the existence of systems associated to a human activity system, which in the majority of cases are designed systems [CHECKLAND 1993, p. 115-118]. Though lacking the provision of a coherent example of a human activity system,[42] CHECKLAND, with his classification of systems, underlines that different types of systems require different methods, and that boundaries between different system types are rarely sharp.

[42] CHECKLAND quotes British Rail and hypothetical experiments considering the exact analysis and prediction of brain activities as examples [CHECKLAND 1993, pp. 115-118].

BOULDING introduces what CHECKLAND calls an “informal intuitive hierarchy” of nine levels of complex systems [CHECKLAND 1993, p. 105]. BOULDING‘s hierarchy ranges from static structures on the first level via more complex mechanisms and control loops to open systems and levels of differently advanced organisms, to finally reach the most complex known systems, the class of socio-cultural systems, on level eight. BOULDING closes his considerations with the “transcendental systems” on level nine [BOULDING 1956], giving credit to the philosophical component of Systems Theory, the far end of second order cybernetics, so to speak.

The vision and ambitious goals of Systems Theory prohibit the exclusion of certain system types. The different attempts to achieve a satisfying classification show the diversity of systems, as well as the difficulty in developing, applying and evaluating concrete approaches suitable for all types of systems. Furthermore, the suggested system topologies lack a distinct differentiation, implying that systems with their inherent dynamic and behavior are usually varied in their origin and properties.

The topology of systems supports the identification of appropriate measures to analyze and cope with a system regarding the inherent properties of a system class. One can differentiate between the used models on one hand, and the applied methods on the other, although both are inseparable in their application. Modeling in Systems Theory often uses mathematical descriptions, considering mathematics to be the most generic language of all. Soft systems in particular are hard to grasp with quantitative measures, which is why more abstract and unspecific descriptions are chosen for that kind of systems. Not without reason, CHECKLAND separates soft Systems Theory and hard Systems Theory, i.e. the “engineer’s contribution” [CHECKLAND 1993, pp. 125ff]. Hard systems, such as technical or designed systems, can be more easily described with quantitative measures once they have reached a sufficient level of maturity, while for both soft systems as well as hard systems in early stages, appropriate qualitative methods are required. Underlying methods or describing models and languages reflect this differentiation and show different approaches in general.

A mathematical description of systems is not uncommon and stems from the prior idea of a unification of science on the basis of the laws of physics. BERTALANFFY establishes a number of mathematical laws to briefly explain fundamental system properties such as growth, competition or centralization, and their applicability for different systems [VON BERTALANFFY 1976, pp. 55-80]. LIN exceeds these considerations and establishes a far more comprehensive framework based on physical knowledge, introducing different existing theories and their application, while pointing out the shortcomings and limitations of the approaches [LIN 1999, pp. 15ff]. Recently, Graph Theory and Network Science, as related mathematical disciplines, emerged from Systems Theory to solve problems related to and emerging from large networks. The mathematical models used are less physical, yet aim for the identification of mathematical laws applicable to networked systems in different disciplines.[43] That mathematics as a language is considered capable of describing all phenomena of the physical world turns out to be a major constituent of Systems Theory, as its

[43] Frequently consulted problems of Graph TheoryGraph Theory are the Königsberg bridge problem or the travelling salesman problem, among others [see e.g. WEST 2001, pp. 1ff].

universality suits the general purpose of Systems Theory, to be applied to systems in general. Other authors acknowledge the importance of mathematics in Systems Theory, but emphasize the relevance of soft systems and the soft systems approach by introducing procedures, principles and methodologies, rather than mathematical models [CHECKLAND 1993, pp. 149ff]. Not only are these approaches meant to solve rather unspecific problems, but they are also intended to be of use to the context of system design in general. Essentially, when entire systems are designed and not only a small part of a product or an organization is considered, the models introduced by soft Systems Theory benefit the design process supporting the architect by aiding his general understanding of the system. Mathematical models and laws are, in most cases, not introduced in combination with a procedure or guideline supporting their application. As a result, researchers in different and diverse areas rarely apply generic mathematical models; rather these are more specific to researchers of general Systems Theory. The rather specific sciences related to Systems Theory, such as Systems Engineering, largely forego generic mathematical models and provide procedural models and a general understanding of Systems Theory as a guideline for application. Mathematical models in the area of engineering are of course numerous, but in most cases it is the specialized laws, findings and calculations that are of relevance for the many challenges in engineering, for example finite elements, fatigue strength or energetic calculations and simulations. For a more general applicability, soft Systems Theory provides models that are generally sufficient for researchers of different areas. As examples for the understanding of soft Systems Theory, CHECKLAND introduced a procedural model derived from action research, an approach originating in the social sciences, which consists of a loop of different phases: expressing the situation in which the system exists; describing the relevant systems for the problem; establishing conceptual models; comparing the models with reality; and identifying and implementing changes (action) [CHECKLAND 1993, pp. 162ff]. The model illustrates what most mathematical Systems Theory concepts are missing, i.e. a clear understanding of how to address a given vague problem in a systematic way. Nevertheless, the model lacks precise methods and specific applicability, making it more of a problem solving process, rather than a concrete solution for specific problems. PULM, in a similar sense, considers Systems Theory as a mindset from an engineering perspective. Among other goals, the focus of his research is not on mathematical laws, but rather on a number of super- and subordinate principles of Systems Theory as its core outcome for practical application [PULM 2004, pp. 48-50], which he applies to the case of the development and design of individualized products, embedded in a procedural model for individualized products. In summary, the models used in Systems Theory are predominately mathematical, while others are abstract descriptions of systems and principles to analyze and design complex systems in general. Both mathematical and abstract models aim for a general system understanding for all kinds of systems.

The models used constitute an important element when discussing existing approaches of Systems Theory. Approaches can further be discussed by the methods or methodologies as an outcome of research in Systems Theory, to be applied in specialized sciences or practice.[44] Additionally, two distinct procedural methods exist, allowing for the research of valid contributions to Systems Theory. These approaches within Systems Theory provide the foundation regarding how to conduct research in the field, and are widely recognized by different authors, such as BERTALANFFY or LIN, who both rely on prior sources [VON BERTALANFFY 1976, pp. 94-99, LIN 1999, pp. 6-7]. Research in Systems Theory can thus be conducted either by observing the phenomena of one particular system to derive noticeable findings and generalizing them to be applied to and validated on further systems (empirical-intuitive), or by observing a large sample size in the beginning, to seek for more generic commonalities defined as axioms of Systems Theory (deductive). While the intuitive approach is considered to be closer to reality and the particular observed system, the deductive approach supplies the mathematically more consistent foundation, but may or may not describe any considered system sufficiently [VON BERTALANFFY 1976, pp. 94-99]. CHECKLAND provides a similar differentiation, but adds the application of Systems Theory in other disciplines as an additional field as a contribution to the study of systems [CHECKLAND 1993, p. 95], from which for example systems engineering emerged. LIN describes a number of approaches (among them, those of CHECKLAND and BERTALANFFY), which differ mainly in terms of the perspective taken, such as human activities, computer-aided problem solving, data analysis, system structures etc. [LIN 1999, pp. 7ff]. As in any other field of science, different approaches to Systems Theory are valid, leading to findings that, in the future, may allow for a consistent entity of Systems Theory. For the moment, the empirical-intuitive approaches tend to be more applicable, providing a general systems understanding, and principles based on that understanding. In combination with the procedural instructions, the derived principles allow for a systematic confrontation with novel problem situations and a purposeful means of addressing the different aspects of the problem system; without Systems Theory, this would not be evident. Subsets of the deductive approach, such as Graph Theory or Network Science, show that this approach provides valuable contributions relevant for both practitioners and scientists.

In the context of the presented research on the management of product architectures, the discussed approaches of Systems Theory provide a number of relevant contributions. The perception of any given subject matter as a system in the definition opens up the possibility of considering different entities, artifacts and influencing elements in product architecture as parts of the overall system. As a result, relevant fields of consideration can widen the scope of product architectures, enabling a more comprehensive perception of product architectures and coping with them. Stating that a system is the “whole” of elements, Systems Theory clarifies the need to thoroughly reflect on which elements are part of the system and which are not. The discussion of Systems Theory further clarified that not only do different views of systems and their complexity exist, but that they require and generate different approaches and methods, underlining the importance and recognition of generic patterns and models

[44] Different authors comment on Systems Theory, criticizing that the findings of Systems Theory are hardly applicable to specific cases without extensive modifications [for a discussion on the topic see LIN 1999, pp. 6-14].

throughout systems of different kind. Finally, the successful application of the results of Systems Theory as such can only be conducted by modifying the approaches to fit the use case in question, suggesting a search for more directly applicable approaches in the more specialized disciplines, which will be discussed in the following chapters.

3.2.2 Operations research

The methods of operations research were developed and applied for the first time during World War II by British and American forces, with the goal of supporting decision-making in complex planning situations [ZIMMERMANN & STACHE 2001, p. 2]. Among the planning problems considered during that period was for example the planning of the composition of ships for transatlantic convoys [DOMSCHKE & DREXL 2002, p. 2]. American scientists in particular continued research on the transfer of the developed methods to business management; these were later subsumed under operations research [ZIMMERMANN & STACHE 2001, p. 2]. The core idea of operations research consists of applying mathematical models to understand, simulate and objectify the rational decision-making process of practitioners [DOMSCHKE & DREXL 2002, p. 1]. Thus, what is currently subsumed as operations research contains methods and solutions dating back to the 18th century.[45]

The overall goal of operations research is to provide a systematic and methodical means for planning in business management, where planning is primarily understood as a mathematical support of decision-making. Operations research thus intends to provide a problem solving process with defined aims: the depiction of a decision problem in reality, by means of mathematical optimization models or simulation models and the application or, if necessary, development of an algorithm to solve the problem [DOMSCHKE & DREXL 2002, p. 1]. DOMSCHKE & DREXL sum up the goals of operations research as modeling, solution finding and the definition of appropriate algorithmic procedures [DOMSCHKE & DREXL 2002, p. 2]. Further characteristic traits of operations research are the striving for optimality, i.e. the maximization or minimization of values, problem quantification and the preparation of decisions. Striving for optimality is not always an option when dealing with real world problems, as the optimum cannot always be clearly defined or reached. Thus, analytical approaches, as well as heuristic procedures, are additional parts of operations research. It is worth noting that, to reach these goals, in-depth information based on reliable existing data is required to enable informed decisions, as would be true for the application of any other type of method [ZIMMERMANN & STACHE 2001, p. 4].

Given the goals of operations research, the subjects under consideration of are numerous. Different examples are regularly mentioned, yet operations research methods are intended to be applied to almost any given system present in industry where economical decisions are made. The relevant areas in which operations research has already been applied successfully, are for example sales, production, purchasing, costing, organization, personnel, or investment. Examples of application can also be found in the technical sector of business, such as in development, design, project management, maintenance or stock keeping. Furthermore,

[45] For early sources of operations research see for example [ZIMMERMANN & STACHE 2001]

examples of operations research are found outside of the business world, as in public administration, the sector of public utilities, public health, the energy industry, or environmental protection [ZIMMERMANN & STACHE 2001, p. 5].

For the wide field of considered systems, mathematical modeling methods are the most common means of modeling. Mathematical modeling requires detailed and reliable information about the considered system to provide equally reliable solutions to the problems under consideration. The models include analytical procedures, approximation procedures, heuristic procedures and simulation processes [ZIMMERMANN & STACHE 2001, p. 4]. The applied simulation processes are predominately stochastic methods, such as the renowned Monte Carlo method [ZIMMERMANN & STACHE 2001, p. 329]. The mentioned models are primarily considered to deliver solutions to a given problem in terms of the explanation of recurring effects or the prediction of future circumstances. DOMSCHKE & DREXL underline the importance of data acquisition to provide reliable solutions. They propose models intended exclusively for information acquisition that serve the description of a system alone, i.e. they depict strictly the real system but avoid causal relations or interpretations. On the other hand, models intended for the explanation of effects or systems include the formulation of hypotheses to explain circumstances that have to be positively or negatively evaluated. Simulation models, as stated by other authors, allow for the prediction of the behavior of systems and future circumstances [DOMSCHKE & DREXL 2002, p. 3].

The general procedure of operations research consists of a process of tasks with steps similar to other existing problem solving processes. The first step of the proposed procedure consists of the abstraction, i.e. in the case of operations research, the depiction of the real problem in a mathematical model. The calculation of the model, the second step of the procedure, provides a solution to the problem depicted in the mathematical model. The interpretation of the calculated results finally allows for a solution to the real problem [ZIMMERMANN & STACHE 2001, p. 3]. DOMSCHKE & DREXL add the detailed recognition and analysis of the problem in the beginning of the procedure, as well as the data acquisition during the process. They further stress the evaluation of solutions prior to the process of interpretation [DOMSCHKE & DREXL 2002, pp. 1-2]. CHURCHMAN et al. consider the control and eventual necessary adaptation of the model, e.g. in case parameters have changed over time, as an additional step after interpretation. Furthermore, the practical realization is considered as a final step, providing personnel with required information, planning of implementation etc. [CHURCHMAN et al. 1961, pp. 23-24]. In general, optimization algorithms of different kinds form the central element of the procedure and serve the overall goal of operations research: the support of decision-making in complex planning situations. Typical examples for methods used in operations research are the critical path planning or network method [ZIMMERMANN & STACHE 2001, pp. 6-47], as well as different methods derived from approaches in Graph Theory [DOMSCHKE & DREXL 2002, pp. 59-73]. Due to the usually complex problems and systems considered, software support is common in operations research to allow for different calculations, react to different circumstances and in general reduce the time of calculations [DOMSCHKE & DREXL 2002, p. 1].

Operations research provides a profound basis for decision-making in complex situations. As mathematical calculations constitute the core results of operations research, it has to be pointed out that quantitative input information has to be available, and the quality of the input information is also crucial. Furthermore, though it is tempting to place blind trust in quantitative results, they must be rationally and thoroughly questioned during interpretation. The universality of the typical procedure of operations research allows for the adaption to numerous problems, as does the generality of the mathematical models. Operations research shows similarities in that generality to the disciplines of Systems Theory and Graph Theory, whose methods and core system understanding it adopted. Summing up, operations research provides widely applicable models and approaches and considers complex situations, in which a more comprehensive view of problems is necessary, as the system as a whole is important in the sense of Systems Theory.

3.2.3 Network Science and Graph Theory

Network Science and Graph Theory, as mentioned in the previous chapters, originally formed a sub-group of mathematics and thus the means of Systems Theory. Due to a large number of commonalities, which are more decisive for the presented work than their differences, both fields of science will be discussed conjointly when possible and differentiated when necessary in the following sections. The commonalities between Graph Theory and Network Science include the approach of modeling the world, i.e. in graphs or networks, the reliance on similar mathematical fundamentals, and the common goal of deriving generalized findings, patterns and principles in the considered networks.

Graph Theory evolved in mathematics in response to apparently simple problems or puzzles still lacking a mathematical explanation [BIGGS et al. 1999, p. 2]. The earliest retrospective mentioning of a graph theoretical problem is that of the Königsberg Bridges from 1736 [BIGGS et al. 1999, p. 3, NEWMAN 2003, p. 169 among numerous other authors], whose solution, the abstract depiction of the bridges and mainland/islands as nodes and edges, posed its accessibility for mathematical considerations which are now formulated as the basis of Graph Theory [DIESTEL 2006, p. 2, WEST 2001, p. 1-2]. The applications, approaches and solutions that followed concerned problems which were increasingly of interest for practitioners and scientists in other areas, most notably chemists, biologists and computer scientists, but also sociologists, whose analysis of social networks dated back to the 1930s [NEWMAN 2003, p. 169]. Nowadays, the influence is far-reaching and expanded to many different disciplines, such as engineering, especially systems engineering, and others [BRAHA & BAR-YAM 2004]. Network theory made use of the mathematical findings of Graph Theory in large networks; the often-cited systems are biological systems, social networks and information networks or artificial but somehow “decentralized” technical structures, such as the world wide web or an electric power grid [NEWMAN 2003, pp. 174-180].[46] The possibility for recent findings in the field is largely due to increased computing power and advanced

[46] For an overview of the findings and examples of Network Science, see for example the rather colorful and popular books of BARABASI or WATTS [BARABÁSI 2003, WATTS 2003], while a more scientific overview can be obtained from NEWMAN and ALBERT & BARABASI [NEWMAN 2003, ALBERT & BARABÁSI 2002], who additionally provide numerous references on the subject.

theoretical understanding [BRAHA & BAR-YAM 2004, ALBERT & BARABÁSI 2002, p. 48], as large networks provide increased data and also follow different laws.

The goals of Network Science differ significantly from Graph Theory, though the science of networks originated from the domain of Graph Theory [ALBERT & BARABÁSI 2002, p. 48]. While Graph Theory seeks to identify axioms and laws appropriate for graphs in general, random networks serve as empirical validation of the results; Network Science seeks to adapt these findings to real networks and thus reduce or expand the number of valid laws under consideration of real world networks [ALBERT & BARABÁSI 2002, p. 48]. The significant difference between the problems typically considered in Graph Theory and Network Science is the size of the networks, which tends to be significantly larger in the examples of Network Science [NEWMAN 2003, p. 169].Where Graph Theory identifies theoretical problems and solutions in often hypothetical environments, Network Science considers the systems of the real world as networks and pursuits the explanation and prediction of the real world by applying and adapting the findings of Graph Theory. In addition to understanding the structure of networks, a central goal of Network Science is to understand the dynamic behavior of networks, such as the development of the world wide web or social networks, which poses a marked difference to Graph Theory [CAMI & DEO 2008, p. 211]. While Graph Theory in itself is a mathematical discipline and, essentially neutral to other fields of science, network theory considers itself to be strongly multidisciplinary [ALBERT & BARABÁSI 2002, p. 48] and observes effects appearing in systems of many different disciplines, not one particular discipline.

Graph Theory and Network Science have different goals, and, as a result, different systems. While Graph Theory perceives systems as rather static structures, Network Science considers primarily the dynamics of systems, such as their growth. Furthermore, in Graph Theory, random networks or hypothetical experiments prevail, as the goal is to increase the mathematical findings, axioms and laws. It turns out that the random networks of Graph Theory lack applicability to the large scale networks considered in Network Science; the real world networks follow different rules, as they are uncontrolled, decentralized, and often rapidly growing, while random networks form uniform structures and require integrity constraints, transactions etc. [NEO & GUPTA 2003]. In contrast, Network Science considers large-scale real world networks, of which social and biological or information networks are the most common examples; the methods of networks science are applied to these examples. In contrast, the consideration of technological networks to current day has been limited to those where the distribution of some commodity or resource is the core functionality of the system, such as electricity or the World Wide Web. The consideration of other technical systems, for example technical products or manufacturing lines, is very rare in Network Science, while the approaches of Graph Theory are rather applicable to problems of this kind. The reason for this lies in the differences of goals and approaches of both disciplines: Graph Theory seeks to identify single nodes and edges of significant meaning, rather than the relevance of properties of the system overall, such as its centrality or connectivity. This is instead the case in Network Science, due to the large networks studied, in which the meaning of a single node or edge is significantly reduced [NEWMAN 2003, p. 169].

Both disciplines use similar mathematical models, but differ in the understanding of their purpose. In accordance with the considered systems, Network Science focuses on laws applicable to real world networks. Graph and matrix representation are commonly used by both disciplines for visualization, as well as for calculation purposes.

The approaches of Graph Theory and Network Science differ above all in their approach to the subject of network analysis. Graph Theory builds its foundation on empirical, artificial and random networks, thus resulting in a large mathematical foundation [see for example DIESTEL 2006 or WEST 2001]. Network Science, on the other hand, observes the dynamics of real world networks and thus reduces or expands the number of valid mathematical laws of consideration of real world networks [ALBERT & BARABÁSI 2002, p. 48]. Graph Theory approaches networks from a strictly mathematical view, which is then applied by other disciplines to their problems. Network Science seeks explanations for the behavior of real world networks, thus setting the starting point of the research on the opposite end from Graph Theory.

The results of Graph Theory and Network Science reflect their different goals and approaches, as was discussed in the previous sections. Summing up, Graph Theory derives axioms and laws for graphs and networks based on mathematical consideration of random networks, resulting in a vast variety. The transfer to real world problems may or may not be possible, depending on the respective systems and disciplines in which they are considered. In contract, Network Science, with the declared goal of understanding and explaining the dynamics of real world networks, seeks interpretations of occurring effects in reality and the desirable generalization to familiar networks from different disciplines if possible.

The relevance for the presented work is given through the situation of product architectures, which themselves can be represented as complex networks, as can their immediate and intermediate environments. Both disciplines, Graph Theory and network theory, explain certain behaviors and properties of networks mathematically, though from different perspectives. While the adaptation of graph theoretical models to the problems in engineering was already discussed and verified by MAURER and others in the context of the design structure matrix [MAURER 2007], the application of the results of Network Science to engineering problems is still absent, with the exception of the given examples of power grids, the world wide web and related networks. One reason for this might be the difference of networks considered in Network Science and technical products, for which only very few metrics apply [see for example SOSA et al. 2005]. The usual technical products are not decentralized and growing through the aspiration of their equal elements, but are defined from the outside to suit a certain purpose.

3.2.4 Systems engineering

The history of systems engineering is closely related to that of Systems Theory, as systems engineering is considered a systems approach and thus correctly set into a close relationship to Systems Theory [MAIER & RECHTIN 2000, p. 8]. Systems engineering is considered the “engineer’s contribution” [CHECKLAND 1993, pp. 125ff] to systems science.[47]

As in operations research or Graph Theory, the first methods, approaches and solutions of systems engineering date back far into the past. In retrospective, ancient undertakings such as the pyramids might be considered as the first applications of comprehensive system approaches in a technical context due to the broad scope of these projects [KOSSIAKOFF & SWEET 2003, pp. 5-6]. World War II is frequently mentioned as having propelled the development of systems engineering means, while the term itself was established in the mid 20th century, when systems engineering positioned itself as a self-contained area of science [KOSSIAKOFF & SWEET 2003, p. 6]. The NASA Apollo missions are perceived to have driven the systems engineering approach even further forward [MAIER & RECHTIN 2000, p. 10]. In the history of systems engineering, the similarities to operations research show in the importance of certain historic events for the development of the scientific approaches, underlining the practical applicability of both areas of research, in contrast to the scientific and abstract means of general Systems Theory or Graph Theory. As discussed in the introductory chapter, the driving forces for complexity and the need for systems engineering are new technologies, competitors, or the numerous specialized disciplines involved, among other [KOSSIAKOFF & SWEET 2003, pp. 6-7].

In the past decades, a few developments put forward in the context of systems engineering are worth mentioning. In particular, those were firstly the definition of architecture frameworks (AF), aiming at the combination of different views of architectures, thus supporting one of the core purposes of architectures and main tasks of systems engineering. Second, the development of the Systems Modeling Language (SysML) in the past years aims to define a milestone for the modeling of complex systems in the area of systems engineering. SysML provides a modeling approach, enriching the Unified Modeling Language (UML) with specific systems engineering views (e.g. requirements), aiming towards the modeling of the relevant architecture entities in systems engineering [SADEK HASSANEIN 2008, p. 75]. The Multiple Domain Modeling (MDM) approach was introduced for the same purpose, i.e. defining a generic modeling approach for complex systems; this provides a simple, yet effective, modeling and analysis approach without claiming comprehensive identification of architecture entities.[48]

Since the late 1980s and early 1990s, so-called architecture frameworks were developed, providing a basis for the architecting of complex systems of a different nature. A number of approaches with origins in the field of information systems [ZACHMAN 1987] developed over time, considering product architectures (for example the Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework [DOD 1997] or the Integrated Architecture Framework (IAF) [WOUT et al. 2010]) and enterprise architectures [SANTE et al. 2007, MATTHES 2011], in many cases

[47] HOLT & PERRY give four different definitions of the term Systems Engineering, comparing different authors [HOLT & PERRY 2008, pp. 2-3].

[48] The Multiple Domain Modeling (MDM) approach will be discussed in chapter 5.4.3

with a strong relation to information systems [compare e.g. SANTE et al. 2007, WOUT et al. 2010]. Recent research proposes the definition of process architecture frameworks by adapting the core principles of architecture frameworks to processes [BROWNING 2009].

The discussion of architecture frameworks will continue more comprehensively in chapter 5.2.1. The approach of SysML as a modeling language for systems engineering will be analyzed in detail in chapter 5.2.2, while the implications and benefits of the MDM approach are presented and evaluated in chapter 5.4.3.

The overall goal of systems engineering is to guide the engineering of complex systems [KOSSIAKOFF & SWEET 2003, p. 3, MAIER & RECHTIN 2000, pp. 5-6]. Systems engineering considers two major aspects of this overall goal: coping with complexity by use of the systems approach, and the guidance through the process of system development as a project management oriented approach [DAENZER 1979, p. 8, KOSSIAKOFF & SWEET 2003, pp. 4-5, ZÜST 1997, p. 28, compare also DAENZER & HUBER 1999]. To enable the management of complexity in both the product and process perspective, as in Systems Theory, a goal of systems engineering must be to consider the system as a whole [KOSSIAKOFF & SWEET 2003, p. 4], i.e. to include the process of systems architecting and integrate the related views. Systems engineering is further striving to support this process by means of a given systems engineering methodology, meant to ensure the goal-oriented and efficient use of the creative potential and expertise of the user of the methodology [ZÜST 1997, p. 24]. The purpose orientation of systems engineering itself is an important aspect of the system of goals of systems engineering, setting the mission of the systems architect into perspective [MAIER & RECHTIN 2000, pp. 10-11]. As a result of the purpose orientation, the systems architect is meant to be an agent of the client, not the builder of the technical system [KOSSIAKOFF & SWEET 2003, p. 4, MAIER & RECHTIN 2000, p. 18]. Systems engineering activities in general are defined to gain insight into the considered system [MAIER & RECHTIN 2000, p. 17]. This insight is not to be considered as a strictly technical approach, but has to fully consider the client, resulting requirements, organization and process.

The systems considered in systems engineering, similar to those of Systems Theory, are diverse and numerous. As a result, different authors provide classifications of systems rather than a precise definition of considered systems. From the perspective of Systems Theory, the systems were already discussed in previous chapters, such as the classification of systems according to CHECKLAND and others, resulting in the differentiation according to the origin of systems, such as natural systems, designed physical and abstract systems, and human activity systems [CHECKLAND 1993, pp. 109, compare also CRAWLEY et al. 2004 and MALIK 2008, p. 219]. These classifications reflect in the perception of systems engineering to encompass the technical as well as the cultural and social aspects of both the system and its complexity. ROPOHL defines three main classes of systems in engineering: the system of goals, the system of action, and the system of objects. In brief, within the system of action, the goals of the system of goals are realized in the system of objects [ROPOHL 1975, pp. 32-33]. The most important system classes from an engineering perspective, according to ROPOHL, include the differentiation into concrete and abstract, natural and scientific, open and closed (or relatively isolated) as well as static and dynamic systems [ROPOHL 1975, p. 32].[49]

To describe the modeling approaches of systems engineering, their importance and role in the process of systems engineering has to be defined. The systems architect is involved in the design process, yet does not contribute to the detail definition during the process directly. The systems architect is therefore strongly dependent on models, substituting the system itself. The models are "acting as surrogates" in the process [MAIER & RECHTIN 2000, p. 163]. Because of the intensively discussed broad range of effects, causes and entities of the product architecture, the models must encompass the many existing different views of the system to enable the systems architect to integrate all relevant constituents of a product to a functional and purposeful system [BOARDMAN & SAUSER 2008, p. 57, MAIER & RECHTIN 2000, p. 163]. The resulting role of the models in systems engineering can be included, as done by MAIER & RECHTIN, as the support of communication, the maintenance of the system integrity (by coordinating design activities), the assistance in design (e.g. by providing templates, organizing and recording decisions), the exploration and manipulation of solution parameters and characteristics (i.e. guiding and recording aggregation and decomposition of system functions, components, and objects), the prediction of system performance, and the identification of critical system elements as well as the provision of acceptance criteria for the certification for use [MAIER & RECHTIN 2000, p. 144]. In brief, MAIER & RECHTIN point out three characteristics of models in systems engineering: first, the role of communication between the stakeholders involved in the process (supporting the maintenance of design, i.e. its integrity and synthesis); second, the multiplicity of views and models; and third, the multidisciplinary, integrated modeling methods, which tie together the various views [MAIER & RECHTIN 2000, p. 164].

KOSSIAKOFF & SWEET give the definition of three types of models in systems engineering, depending on how the system is modeled. These types include schematic or descriptive models (e.g. organization charts or data flow diagrams), mathematical models (e.g. statistical distributions, differential equations used in system dynamics) and physical models (direct reflection of the physical characteristics of the actual system or parts of it, such as physical prototypes or mock-ups) [KOSSIAKOFF & SWEET 2003, pp. 410-411]. To complete the picture, MAIER & RECHTIN give a classification of the models consisting of six types, depending on what the model depicts: a model of the purpose or objective of a system (e.g. customer demands); the form of the system (e.g. physical model of the system); behavioral or functional models of the system; a model of the performance objectives (e.g. the effectiveness of fulfilling the technical requirements); data models (e.g. information retained in the system and its interrelationships); and managerial models (e.g. process charts, workflow models) [MAIER & RECHTIN 2000, p. 146].

[49] It has to be noted that a classification of systems, to become more than an end in itself, has to result in reasonable approaches and models in accordance with the classification, so that the classes reflect certain models and procedures to cope with the particular type of system. The following chapter will discuss this subject by the example of product architectures.

As coping with complexity is the overall goal of systems engineering, there are naturally numerous different methods relate to that goal. As first approximation, the approaches and methods of systems engineering can be divided into modeling methods and process-related methods, together forming an “architecture framework” [MAIER & RECHTIN 2000, pp. 221-234]. A characteristic of most methods of systems engineering is their focus on qualitative rather than quantitative knowledge, especially in the early phases of design and when dealing with new technologies [KOSSIAKOFF & SWEET 2003, p. 4], as is the case in most systems engineering projects. The number and outcome of methods are numerous, and some will be discussed in detail in the following chapters; at this point of this work, the focus is on grouping methods according to their purpose within the systems engineering process. A cogent structuring is provided by what is called the “essentials of systems engineering” by BOARDMAN & SAUSER [BOARDMAN & SAUSER 2008, pp. 47-61]. The authors sum up into seven categories the crucial areas of support by systems engineering methods; the previously discussed goals and models of other authors uphold these categories. The first aspect of method support is described as lifecycle recognition, i.e. the recognition of the temporal dependencies of technical systems, which includes the contextual and stakeholder dimensions. The authors stress the fact that not only the product lifecycle has to be considered in systems engineering, but also the lifecycles of production technology, organizations, knowledge, technologies etc. [BOARDMAN & SAUSER 2008, pp. 48-50]. As systems architect, the tensions between the top-down approaches of project management and control and the bottom-up “project reality” have to be matched. Both are necessary and beneficial for the overall project, yet the discrepancy between both has to be bridged [BOARDMAN & SAUSER 2008, pp. 50-52]. The authors further mention the ambiguity and vagueness of requirements and their interrelation and dependence from the solution space as a major field of method application [BOARDMAN & SAUSER 2008, pp. 52-53]. As became apparent during the discussion of the models of systems engineering, the integration of stakeholders, their different viewpoints and methods to a coherent methodology is of great importance for the systems architect [BOARDMAN & SAUSER 2008, pp. 53-55]. The process of decision-making, including the identification of feasible candidates, formulation of criteria, weighting of performance and selection making, requires method support as well, for example through a trade-off study [BOARDMAN & SAUSER 2008, p. 55]. To support the modeling and simulation, a family of models is required, as was previously discussed [BOARDMAN & SAUSER 2008, pp. 55-58]. Finally, the operational effectiveness has to be considered to ensure the long term perspective and sustainability of both the technical system and the process [BOARDMAN & SAUSER 2008, pp. 58-61].

The outcome of successful systems engineering projects reflects the numerous models and methods of systems engineering. The systems architect has to deliver a representation of the systems architecture. The representation includes an abstract design of the system, which is usually not enabling the immediate build up of the technical system. Indeed, the system has to be refined. In accordance with the essentials of systems engineering, the results must contain, not only the physical representation of the system, but also information about cost, behavior, performance, human organization etc. [MAIER & RECHTIN 2000, p. 18].

In summary, the relevance of this research is confirmed through the existence of technical systems primarily. Systems engineering takes into account the uncertainty of the early phases of the development of technical products by providing qualitative means and procedures that aim to guide the complex process that is product development. This also is one of the goals of the presented work. The systems engineering process requires an encompassing and comprehensive approach to the product architecture. The results have to regard the relevant lifecycles, bridge the tension between the top-down and bottom-up approaches, provide different viewpoints and support decision-making, as well as modeling and simulation, and finally provide the support of the long-term perspective. These considerations, in combination, should guide the thoughts and results given in this work and, in the end, help to evaluate how far these have been achieved and are relevant for the presented work.

3.2.5 Engineering design research and design theory

Similar to systems engineering, research in engineering design experienced a boost in the early 20th century, when the characteristics and principles of engineering design recognized since the 19th century were systematized and documented in step-by-step approaches [PAHL et al. 2007, p. 11]. Although an urgent need for the improvement of products and processes was previously identified, World War II boosted the efforts for efficient products and processes. Systematic thinking and a methodical approach to design was further propelled by staff shortages in the 1960s [PAHL et al. 2007, p. 12]. Since then, different, increasingly comprehensive approaches evolved over the years under numerous scientists. Some of these approaches are still used and valid as part of more recent research [PAHL et al. 2007, pp. 13-14]. The increasing scope and multiple disciplines of products called for these sophisticated approaches [WIESE & JOHN 2003, p. 55]. The different phases that engineering design research went through in the 20th century can be described by three stages. The early works are characterized as experiential, largely based on the documented experience of renowned engineers. The intellectual stage, generating systematic approaches, started in the 1960s, while experimental approaches in design were developed in the 1980s [BLESSING & CHAKRABARTI 2009, p. 3]. Today, all types of research characterized by these stages are conducted and exist in parallel. The clear definition and distinction of approaches appear to be melding at times [BLESSING & CHAKRABARTI 2009, pp. 6-7]. The major influences on design science, according to HUBKA & EDER, are philosophy, psychology and sociology, work science, mathematics, cybernetics, information science, management, and invention theory, among others [HUBKA & EDER 1996, pp. 89-93]. It is in part due to these influences and the resulting diverse backgrounds of researchers, that a precise, consolidated and acknowledged view on design research is difficult [BLESSING & CHAKRABARTI 2009, p. 4].[50]

It is the overall goal of engineering design research to collect and classify knowledge about design in order to obtain consensus about engineering design, and furthermore to bring this knowledge to use, i.e. understanding and supporting product design in terms of the

[50] Brief historic overviews on engineering design can be found in BLESSING & CHAKRABARTI, PAHL et al. and HUBKA & EDER [BLESSING & CHAKRABARTI 2009, pp. 2-6, PAHL et al. 2007, pp. 10-14, HUBKA & EDER 1988, pp. 4-7]. For a more elaborate discussion see HUBKA & EDER [HUBKA & EDER 1996, pp. 49-66].

improvement of products and processes [HUBKA & EDER 1996, p. 36, BLESSING & CHAKRABARTI 2009, p. 5]. The understanding is an especially important issue, in order to be able to support and improve design processes and designs [BLESSING & CHAKRABARTI 2009, p. 5]. A differentiation of three goal areas can be made: engineering design research with the goal to improve the practice of design in industry; to contribute to design science, i.e. answering scientifically relevant questions and positioning design science in the area of science overall; and research to improve the education of design, i.e. enabling quicker and better education [HUBKA & EDER 1996, pp. 74-75]. HUBKA & EDER name examples of a number of goals to be derived from the mentioned overall goals, such as generating of optimal quality of products, reducing of design times, reducing risks, reducing human routine work during design, enabling and improving computer application in design etc. To give a more generic view on design science, HUBKA & EDER describe the "demands of science" [HUBKA & EDER 1996, p. 75], stating that research in design (and research in general) has to be purposive (identification of a problem worth researching), inquisitive (seeking to acquire new knowledge or new relationships among knowledge elements), informed (conducted based on an awareness of previous research), methodical (planned and carried out in an efficient and disciplined manner), and communicable (testable and accessible results) [HUBKA & EDER 1996, p. 38].

In accordance with the goals in design research, the product and the design processes are the main subjects of consideration [HUBKA & EDER 1996, p. 82, BLESSING & CHAKRABARTI 2009, p. 5]. Models exist both for the areas of design research, products and processes and usually stand in close relation to a method or methodology with a given purpose. In the following chapters, product models will be discussed more intensively, while process models are of marginal importance for the presented work. FUCHS gives an overview on numerous methods and the characteristics of the models used with their application, among which are predominately different functional diagrams and process models [FUCHS 2004, pp. 152ff]. He further establishes a classification based on the respective content (of models as functional models, principle models (model of working principles), component models and process models), structure (morphology, relation, taxonomy, transformation) and purpose (quantitative, qualitative) [FUCHS 2004, pp. 15ff]. An extensive overview on process models is also introduced by KUSIAK or KREIMEYER [KUSIAK 1999, pp. 2ff, Kreimeyer 2010, pp. 264ff.].

On an abstract level, methods and approaches of engineering design research can be divided into three major categories, according to HUBKA & EDER. The first category is research into design, which can be roughly outlined as observations for understanding design and its nature, as well as the design process. BLESSING & CHAKRABARTI describe this type of research as experimental. Research for design as a second category in engineering design includes the creation of tools, often computer-based, design methods and forms of modeling. BLESSING & CHAKRABARTI describe research for design as intellectual. As last category, research through design, i.e. abstraction through self observation during designing, hypothesizing, and testing is described by BLESSING & CHAKRABARTI as experiential [BLESSING & CHAKRABARTI 2009, p. 3, HUBKA & EDER 1996, p. 38].

To discuss the outcome of engineering design research, its goals, approaches and models of design research need to be combined, resulting in a multidimensional space of possibilities. Due to the multiple facets of design research, these topics could only be roughly categorized, rather than described exhaustively in the preceding paragraphs. To conclude the chapter with the outcome and results of engineering design research, the following outline by PULM gives a reasonable overview on possible outcomes; however, due to the diverse categories in design research, this cannot describe the outcomes comprehensively. The brief discussion beforehand should enable the reader to get a picture of research in engineering design, while the following list concludes the discussion in a reasonable, yet incomprehensive manner, as not all facets can be considered. PULM sums up the results to which design research aspires as: methods and strategies, results of fundamental research in the areas of empirical research (sociology, psychology), innovation, creativity and design theory, coordination of organizations and processes, computer support (intelligent systems, tools, models, design-automation), different Design for X aspects and education [PULM 2004, p. 76].

The relevance of engineering design research for the presented work is significant. First of all, the presented work seeks to provide a relevant contribution to the field of engineering design research, as well as to systems engineering. Secondly, the models, methods and approaches developed in engineering design research are highly relevant for a comprehensive approach to product architectures. They will be discussed in the following chapters, acknowledging that a complete overview can hardly be achieved in a work such as this. As a third conclusion, the scope of engineering design research, will help to clarify the position of the presented work within the field as well as its value to the area of science.

3.3 Summary

The previous sections defined the understanding of complexity and discussed the related fields of research. As an outcome, a sound understanding of complexity was reached, and the relevant fields of research identified, parts of which will be discussed more intensively in the following sections. It became clearer that complexity as a challenge has not been ignored in science. The following sections will discuss the feasibility of concepts and approaches.

It is worth mentioning that the areas of business management and software engineering, both equally broad and diverse as engineering design research, also provide valuable contributions to the topic. A reasonable overview on the engineering relevance of software engineering is given for example in [KOSSIAKOFF & SWEET 2003, pp. 361-408]. Individual approaches and methods from those areas are referred to and discussed in the following sections, when appropriate.

4. Product architecture model and domains

The previous chapters laid out the situation in which product architectures are embedded and defined both the origins, as well as the character, of the complexity that causes the need for an intensive discussion on the topic. The general challenges of the management of product architectures were derived based on that situation, resulting in a number of goals for research in the area of the management of product architectures. To address these challenges in accordance with the established research method, the following paragraphs discuss the character and constituent parts of product architecture in detail. Together, they form the basis for a comprehensive product architecture model. As a result, the last chapter of this section proposes a framework for product architectures, incorporating the outlined constituent parts of the product architecture, as well as the relevant dependencies, as a basis for the following discussion of methods and approaches.

4.1 Scope of the product architecture

Dealing with product architectures requires a broad understanding not only of the term "product architecture" itself, but also of the many related aspects of it. It is the purpose of this chapter to deliver a definition of considerations for a comprehensive approach for the management of product architectures. To form a common basis, the brief definition of CRAWLEY et al. sums up the core perception of product architecture on an abstract level, stating that the "systems architecture is an abstract description of the entities of a system and the relationships between those entities" [CRAWLEY et al. 2004]. While CRAWLEY ET AL. state that some systems can be represented quite completely by networks (relying on the works of e.g. BARABASI and WATTS), this networked view represents just one property of systems, though a very important one [CRAWLEY et al. 2004]. The goal of the following paragraphs is to discuss further the entities of the product architecture as a network and how their relationships can be described.

Although some authors define product architectures on the basis of their physical components alone [HUBKA & EDER 1988, p. 69, RAPP 1999, p. 9, SCHUH 2005, p. 73], it is mostly the interrelation of physical components and functions that is considered to be product architecture [see e.g. PIMMLER & EPPINGER 1994, ULRICH 1995, SUH 2001, p. 11, CRAWLEY et al. 2004, BONJOUR et al. 2009]. Frequently cited is ULRICH's definition "The architecture of the product is the scheme by which the function of the product is allocated to physical components." ULRICH specifies the arrangement of functional elements, the mapping from functional elements to physical components and the specification of the interfaces among interacting physical components [ULRICH 1995]. BAUMBERGER similarly concludes by summing up the different definitions, and defines the product architecture as the functional, structural and hierarchical relations of the product and its constituent parts [BAUMBERGER

2007, pp. 99-100, compare HANDKE 2000, p. 29].[51] The following paragraphs will discuss the constituent parts, based on relevant literature, focusing on the discussion of product architectures, and concluding with a picture of what is understood as product architecture in this work.

According to the definitions, the most important, significant and obvious entities of product architecture are the physical constituents. By giving two examples, the class of physical constituents is detailed, providing the inherent hierarchy of physical constituents or its possible versions. WYATT et al. identify modules, components and key parameters as the upper levels of product architecture, giving a reasonable structuring of the physical entities of the product. Other than the physical view, WYATT et al., among others, consider functions as equally relevant entities of product architecture [WYATT et al. 2008]. To detail the physical architecture entities more thoroughly, HANDKE provides a hierarchy derived from a literature review encompassing different types of machines. Bottom up, the hierarchy starts with geometrical features (ranging from 1 to 3-dimensional features), continues with single parts and different levels of their assemblies (e.g. sub- and main-assemblies) to machines and units. HANDKE continues the listing for the field of production equipment with production lines, plants and different groupings thereof [HANDKE 2000, p. 67, see also PAHL et al. 2007, pp. 27-28].

To complete the physical perspective of product architectures, a classification can be made in accordance with the composition of mechatronic systems, i.e. mechanical, electrical and software elements as well as feedback control systems [compare FELGEN 2007, p. 42-47, GAUSEMEIER et al. 2001, VDI 2206, p. 14]. FUKUZAWA gives the illustrative example of mechatronic systems by multifunction printers, focusing on the domains of hard- and software, as well as functions and different software hierarchy levels [FUKUZAWA 2008]. SADEK HASSANEIN furthermore underlines the importance of services as an integral part of products, which tend to increase or even partially replace traditional products in the future [SADEK HASSANEIN 2008, pp. 5-11]. By the given definition [SADEK HASSANEIN 2008, p. 25], the solution developed by the provider, i.e. the company, consists not only of the physical parts of the product, but allows for the fulfillment of the required functionality through the combination of physical and immaterial elements, intended to increase the flexibility and upgradeability or adaptability along the lifecycle. To fulfill the user's required functions, service elements are treated similarly to physical elements. A more thorough classification of the physical and service entities of the product is frequently derived from the functions provided by the respective entities. In the spirit of a concise segregation of the product architecture entities, the discussion of different functions will be discussed in the following paragraphs.

The functional perspective of the product delivers many views and perceptions. In science in general, and in cooperation with industry especially, the term "function" does not always receive the same treatment. It is principally the degree of solution neutrality that differs across the various definitions. Whereas in industry functions are normally considered as a

[51] BAUMBERGER uses the term "product structure" in his work [BAUMBERGER 2007], which is considered equivalent to "product architecture" throughout the presented work.

combination of physical elements, defined by De Lit & Delchambre as “functional subsets” [De Lit & Delchambre 2003, pp. 105-109], researchers claim that the close correlation to physical elements prevents the definition of novel solutions, which is why Lindemann asks for a solution neutral definition of functions [Lindemann 2009]. Steinmeier discusses different perceptions of the term “function” in the context of a systems approach to product development [Steinmeier 1999, pp. 73ff]. In an attempt to structure the discussion of Steinmeier [Steinmeier 1999, pp. 73ff], the following characteristics allow for a classification of functions: the function’s purpose (transformation of input to output, desired states or behavior, influence on other functional elements), the chosen level of abstraction (due to the chosen system boundaries, constricting boundary conditions, relation of the considered system to the superior system) and the system purpose (main, secondary and harmful functions). The function’s purpose describes the intended effect the function or part has to fulfill, such as the transformation of a material, signal or energy flow, or the enabling of another function by changing their inherent behavior. The chosen level of abstraction influences the ability of solution-neutral formulations of functions. While the function of an automobile in the overall social context enables the independent travelling of individuals, the technical function would be to transfer stored energy of an undefined type to rotational energy of the wheels. Boundary conditions might specify the stored energy (e.g. chemical, electrical) and thus reduce the degree of neutral functional descriptions. The system purpose divides the functions into three groups: functions necessary to fulfill the overall purpose of the system (main functions); supporting functions required to enable the main functions (secondary functions); and harmful functions existing due to unwanted but unavoidable side effects of the main and secondary functions. Crawley et al. define the main functions of the product as “primary functions” and differentiate them from “ilities” such as durability, maintainability, flexibility etc. [Crawley et al. 2004], which by other authors are considered in the context of Design for X [compare Hemel & Keldmann 1996, p. 73, Huang 1996, p. 3]. De Lit & Delchambre largely agree with the given entities, yet add “functional subsets” as a category for components which in combination fulfill at least one function of the integral product. As such, the functional subset or subassembly provides a similar classification to the assembly, yet from a functional, not strictly physical, view [De Lit & Delchambre 2003, pp. 105-109]. Levis states that different architectures are required during product concretization, similar to Ponn & Lindemann, and relies mainly on functional and physical architecture [Levis 1999, Ponn & Lindemann 2008]. As summed up by Crawley et al., Levis differentiates between physical, technical, and dynamic operational architecture. The physical architecture is composed of at least a graph or matrix representation of physical constituents and interrelations. The technical architecture details the physical architecture using a set of rules to achieve the requirements of the product, while the dynamic operational architecture depicts the behavior of the product over time [Crawley et al. 2004, p. 5, Levis 1999].

In addition to the physical entities and system functions, the so-called working principles of the product greatly contribute to the product properties during the process of product concretization [see e.g. Pahl et al. 2007, pp. 38ff, Ponn & Lindemann 2008, pp. 75ff or Handke 2000, p. 29]. The definition of working principles allows for the definition of technical principles to fulfill the desired requirements and functions of the product. Yet, working principles are detached from a precise geometrical and physical representation of the product and thus allow for bridging the gap between the functional description of the product architecture and its physical components. With product design being the activity where physical effects play their most important role, the physical effects, too, are a matter of standardization or innovation in the manufacturing firm, e.g. to improve manufacturability, productivity or technology offers to the market.

KUSIAK identifies requirements other than functions and components as the core entities during conceptual design [KUSIAK 1999, pp. 201ff]. BONJOUR similarly sums up the previously discussed entities of the product architecture, from the perspective of variant management as requirements (customer expectations and lifecycle requirements), functions (functional architecture), and physical (or design) architecture (subsystems and components) [BONJOUR et al. 2009]. BONGULIELMI differentiates between the views of the product architecture, reaching the same conclusion, namely customer (requirements) and technical view (functions and components). According to BONGULIELMI, both are necessary and sufficient to capture knowledge relevant for example for configuration decisions in variant rich design [BONGULIELMI 2003, pp. 61-63]. TRIPATHY & EPPINGER provide an example of a systems architecture model in which they incorporate the following types of requirements: (technical) product requirements, product performance specifications, and requirements to the industrial design (i.e. the look and feel of the product) [TRIPATHY & EPPINGER 2007]. HANDKE structures requirements according to the product entity to which they apply, namely the product family, product modules, product functions and different tasks, i.e. the rules to follow when building a system [HANDKE 2000, p. 41].

The product components, inherent working principles and functions in combination result in the system's properties, intended to fulfill the identified product requirements. WEBER differentiates properties from characteristics for the Characteristics-Properties-Modeling (CPM) method, depending on how influenceable they are [WEBER 2005a]. From this perspective, characteristics, properties, required properties and the relations between characteristics, properties and external conditions are all relevant. The system characteristics, such as structure, shape and material, are then directly influenceable by the designer; on the other hand, the properties, such as weight, safety and the different existing "ilities" [WEBER 2005a], describe the product's behavior and are not directly influenceable by the designer. BERNARD divides product properties into classes depending on the available knowledge about the respective properties. As a result, BERNARD provides four classes of properties: *validated properties,* known in the early phases of design due to carry-over or bought-out parts; *real properties*, about which knowledge is gained throughout the process;[52] *reliably predicted properties*, known through methods and simulation models; and *unknown properties*, unidentified in early phases due to wrong predictions and assumptions [BERNARD 1999, p. 29].

In addition to the entities mentioned above, some authors name further entities that are necessary for the approaches presented, due to the use cases studied. DEUBZER et al. provide a

[52] The amount of *real properties* increases throughout the design process, gradually displacing both *reliably predicted* and *unknown properties* at the end of the process, when the product is completely designed and all properties are known [BERNARD 1999, p. 29].

use case of a comprehensive approach to variant management that incorporates the views of both design and sales department. In addition to the discussed entities of functions and components, the authors consider a further entity in their product architecture model: *meta-data*, such as part numbers, relevant to estimate product cost, as one domain of the model. Further domains include the different *equipment* from which customers can choose (optional and necessary equipment or packages), the existing *segments* and *product types* by which the products can be classified, as well as the *product lines*, which inherit different product types for different segments [DEUBZER et al. 2008]. SCHUH discusses the entities of a product family in a similar fashion, presenting a comparable use case and considering the product family as a relevant entity of the product architecture, summing up the entities such as packages, product lines etc. [SCHUH 1989, p. 29]. SANDER, on the other hand, establishes a framework for a comprehensive library for solution finding, identifying the different use cases and application scenarios, apart from the necessity of functions, working principles (or effects), and solution elements (components) [SANDER 2001]. The inclusion of organization or "meta-data" use cases or application scenarios is especially important when strategic decisions are necessary. Possible decisions include the comparison of design alternatives and judgements about product platform or product family programs, including their design and economic value and impact. In that sense, the integration of organizational entities of the product portfolio and organizational maintenance into the product architectures appears to be highly relevant for the decision-making processes during design, thus requiring a comprehensive approach, while differing from other product architecture entities.

In contrast to the entities discussed above, the following and final considerations provide a rather unconventional view of the product architecture. While the entities mentioned earlier represent the product itself through differentiated perspectives of the product architecture such as physical views, functions, requirements or organizational means (i.e. inherent meta-data), several authors add a further dimension to the product architecture that does not represent the product architecture, but rather the implications considered during the process. Although CRAWLEY et al. define architecture as arrangement of entities, as discussed in the previous paragraphs, they bring forward a second view, i.e. the "rules to follow when creating a system" [CRAWLEY et al. 2004]. GULATI & EPPINGER define product architecture similarly, i.e. as a "set of technical decisions (the plan) for the layout of the product, its modules, and for the interactions between the modules" on the one hand, but follow the definition of ULRICH as well, acknowledging both the view of entities and rules to follow when creating a system [GULATI & EPPINGER 1996, ULRICH 1995]. YASSINE & WISSMANN define design rules as standardized interface parameters and protocols [YASSINE & WISSMANN 2007], which provide reasonable examples of design rules.

4.2 Modeling product architectures

The model of the product architecture and its respective entities has to encompass the results of the discussion in the previous chapter. The task of modeling the product architecture, including the abovementioned entities, must be executed in the sense of systems engineering (or systems architecting). A model in general is a reproduction of reality suitable for this context [FUCHS 2004, p. 18], while the appearance and content of models in general depend on the context in which the user of the model requires information [DAENZER 1979, p. 13, PAHL et al. 2007, p. 28-29]. Additionally, the description (or model) of a system largely influences its appearance or perception as an either complex or simple system [SIMON 1962]. The model of the product architecture does not aim to replace all other existing representations, but rather to allow for a comprehensive perception of the product architecture. As a result, the product architecture model establishes a profound basis for decision-making during the early phases of design. To approach the required model suitably, the following sections discuss the requirements to a systems modeling approach, as well as existing classifications of models. A modeling approach is proposed, fulfilling the discussed requirements and providing a suitable basis for further considerations in the sense of a comprehensive product architecture management.

In general, a model of the product architecture provides the **documentation of artifacts or entities of the product architecture**, each with a given purpose suitable for the situation or task to be executed with the aid of the model. The content of the model depends not only on the tasks to be executed or the current situation, but also to a large extent on the stage in which the process is and the resulting level of abstraction. The model of the product architecture might therefore range from requirements lists in early phases to geometric models in the later phases, which themselves evolve from sketches to simulation models.

The goals of a model of the product architecture and the inherent requirements can be derived from the goals of the work presented in chapter 1.2, the understanding of systems architecting presented in chapter 3.2.4 and the discussion of the product architecture itself in chapter 4.1. It is the overall goal of the model of the product architecture to support both the process of architecting as well as architecting itself [CRAWLEY et al. 2004, p. 9]. The detailed goals can be derived as follows.[53]

As numerous approaches (i.e. principles, methods and tools) exist for the **different tasks and aspects of systems architecting**, a supporting model for a comprehensive approach is required to consider their existence. As the history of systems science has shown, it is not possible to establish a method or model substituting the existing methods and models and incorporating all of their different viewpoints and outcomes. It is nevertheless necessary to acknowledge existing and recently researched approaches, to allow for the demand of comprehensiveness. The approaches have to be represented in terms of the respectively considered product architecture entities and interdependencies. As a result, the outcome of the application of an approach can be pictured in the product architecture model. Based on this information, the influences on other entities, and thus the results of other approaches, can be identified and analyzed. In doing so, the product architecture model can provide the applicability of existing approaches, as well as the integrity and continuity of the different approaches supporting the process of systems architecting. As an example, the results of FMEA in terms of failures of components reflect on affected functions and point to available alternative solutions in the solution space in both the functional and component domain.

[53] The following discussion tries to elaborately transfer the findings of different authors into the context of this work [BOARDMAN & SAUSER 2008, p. 57, CRAWLEY et al. 2004, p. 9, KOSSIAKOFF & SWEET 2003, pp. 410-411, MAIER & RECHTIN 2000, pp. 144-146 and 163-164].

Following the argument of the integrity of existing approaches, the architecture model has to **encompass the different entities** accordingly, thus tying together the various views with the product architecture. The multiplicity of models necessary for systems architecting is thus tied to a comprehensive model, with the coupling of different entities across domains, as well as levels of concretization or abstraction. As such, there is interrelation between different functional models, such as hierarchical or networked models, as well as between the different considerations of the physical architecture, such as product families and physical effects.

Different existing views on approaches and models are relevant because of the numerous stakeholders involved in the process of systems architecting, in brief the different organizational entities and customers or clients inside and outside the company. It is thus an important role of the product architecture model to **establish feasible means of communication across the different views**, enabling the understanding of different stakeholders with one another, and resulting in a vast amount of information, based on which the systems architect is able to reach conclusions and decisions. To enable communication, the model of the product architecture must be based on rather neutral techniques, understandable by different people and unbiased with respect to the different professions or specialties involved. As such, the discussed modeling approaches have to provide this neutrality and ability to support communication. An additional requirement stems from the necessity for the architect to be aware of the different perspectives and models. The systems architect has to be aware of these viewpoints, and the comprehensive model has to support this awareness.

With the different stakeholders comes their involvement in the process of design, which leads from conceptually abstract to detailed. This **continuity of stakeholders** reflects in the models and approaches used, leading to a two-dimensional evolution of the entities of the product architecture. As a first dimension, the entities of the product architecture are refined during the process, and as such underlie a continuous modification. As an example, the physical domain ranges from physical principles to geometrical descriptions of components, which are also increasingly refined up to the end of the process. Additionally, different domains are relevant during the process, requiring a transformation of entities from one domain to another, for example from requirements to functions, and on to physical principles. This two-dimensional variation further complicates coping with product architectures, due to the interconnectivity of its entities and the occurring changes of the system over time.

The purpose of the mentioned requirements for the modeling technique lies within demands stemming from the use of the model in the early phases of design. As a result of that position in the process, the **support of design synthesis** is very relevant, as is the **consideration of the product lifecycle** during that phase. Synthesis is the main task in the early phases of design, and requires the logical incorporation of the later phases of the lifecycle, to allow for consistent product family strategies or the enabling of maintenance, recycling etc. From that core idea stems the requirement for a comprehensive and continuous support and integration of methods, models and tools. It is the aim of the model to provide templates and principles, which assist the design process. Highly relevant for design synthesis is the establishment of a comprehensive solution space, allowing for the exploration of possible solutions and the identification of alternatives through the manipulation of characteristics and properties. In the sense of systems engineering, these procedures can be described as the support of the solution finding process by inducing and documenting the iterative occurrence of analysis and synthesis. Parallel to design synthesis comes the evaluation of systems and alternatives the decision-making processes of analysis. Especially in the early phases, the identification of system properties and system performance is marked by strong traits of prediction, rather than analysis. The gradual support of system synthesis and analysis across different entities of the product architecture has thus to combine the knowledge of different levels of detail, e.g. known reused subsystems and subsystems currently under development. The identification of properties during analysis is aligned with the identification of critical system elements, which can be critical functions, components or requirements. Identifying system elements as critical may be necessary due to the development risks of components, due to cost or market dynamics, conflicting goals or elements causing undesired properties or behavior etc.

The fulfillment of the previously mentioned requirements must enable the model to support the systems architect in coordinating design activities, making appropriate decisions, communicating with stakeholders about their requirements and further lifecycle requirements, executing design synthesis and analysis etc. The model should further allow for continuity and consistency by interrelating all relevant entities of the product architecture, as well as their different levels of detail.

The requirements for the modeling of product architectures can be summed up as follows:[54]

- Documenting of product architecture entities
- Supporting the tasks of systems architecting by incorporating interfaces into existing approaches
- Incorporating the different product architecture entities
- Supporting communication among stakeholders
- Enabling the continuous involvement of stakeholders by considering the detailing of entities
- Supporting design synthesis
- Considering the product lifecycle

Different classes or types of models can be discussed regarding their suitability for a product architecture management approach. A common classification of models proposes the classes of graphical, tabular, textual and analytical models [see e.g. FELGEN 2007, p. 33, compare FUCHS 2004, pp. 93-94, GÖPFERT 1998, p. 22, HOLT & PERRY 2008, p. 20]. KOSSIAKOFF & SWEET differentiate between schematic and descriptive models (e.g. organization charts or data flow diagram), mathematical and analytical (e.g. statistical distributions, differential equations used in system dynamics) and physical models (i.e. direct reflections of the physical characteristics of the actual system or parts of it such as physical prototypes or mock-ups) [KOSSIAKOFF & SWEET 2003, pp. 410-411]. Physical models often consist of a combination

[54] Based on the findings of BOARDMAN & SAUSER 2008, p. 57, CRAWLEY et al. 2004, p. 9, KOSSIAKOFF & SWEET 2003, pp. 410-411, MAIER & RECHTIN 2000, pp. 144-146 and 163-164

of graphical and analytical modeling, such as CAD-models, while descriptive models can be represented in graphical, tabular or textual form. A system can usually be modeled in different ways, some of which prove to be more useful in certain situations than others. Product models, for example, exist in all different forms. Requirements lists model the product in tabular form, functional descriptions are textual, CAD-models and manufacturing drawings pose graphical representations of the product while numerous analytical approaches support product analysis and synthesis, such as estimated calculations or virtual prototyping. Not only can different entities be modeled in different ways, but also the same entity, such as product requirements, can be modeled in a variety of manners. The change of perspective is not only necessary, but even regarded as supportive for solution finding processes. Different procedures and models actively incorporate that principle to find new solutions, which are generated on the basis of existing solutions [KNOBLICH 1997, p. 214-215].

For the desired approach, where communication and integration of different models are to be achieved, textual and analytical models are traditionally not feasible. Visualization and understandability of the models are largely relevant, making textual and analytical models too time consuming to establish as well as understand and discuss. Nevertheless, it is essential to at least provide linkages to analytical models, in order to incorporate results from analytical models properly. Textual descriptions, on the other hand, are useful at any time as an explanation of documented elements.

To further structure the model of the product architecture within the context of a comprehensive approach, MAIER & RECHTIN provide a sound classification based on the content of the model, which in the following section is grouped in accordance with the differentiation of ROPOHL into goal-, object- and action-system [MAIER & RECHTIN 2000, p. 146, ROPOHL 1975, pp. 32-33].

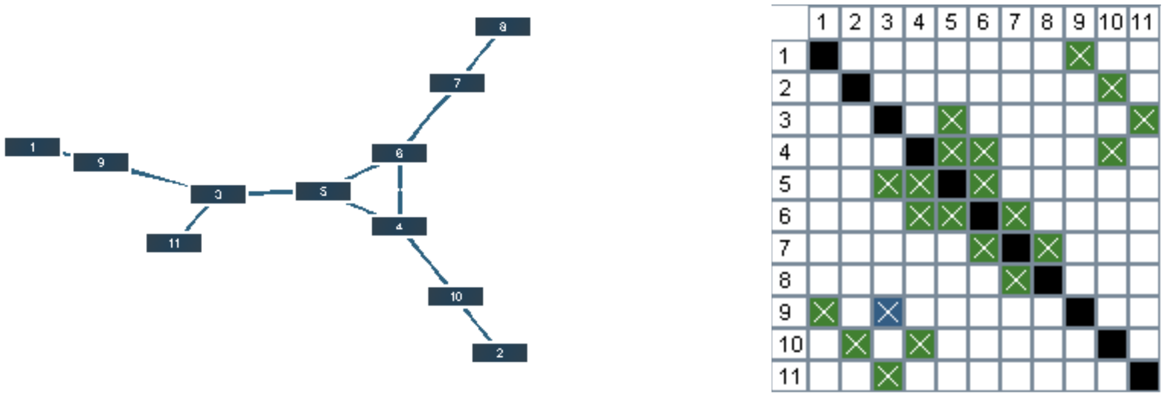
In the context of the goal-system, MAIER & RECHTIN identify the purpose or objectives as a significant class, depicting what the client wants. The goal system is completed by performance objectives or requirements, describing how effectively the system does fulfill its purpose. The object system can be divided into the four following modeling aspects: form (geometry, depicting what the system is); behavioral entities; functional entities; and finally data, i.e. the information retained in the system and its interrelationships. The action system contains the managerial aspects of product architecting, i.e. the process by which the system is constructed and managed [MAIER & RECHTIN 2000, p. 146], or the rules defining the product; respectively, the set of decisions resulting in the product [CRAWLEY et al. 2004, GULATI & EPPINGER 1996].

The models of each entity type of the product, such as requirements, functions or components, can be classified further using means described by SADEK HASSANEIN as the degrees of formalization, concretization and detail [SADEK HASSANEIN 2008, p. 99, compare also FUCHS 2004, pp. 71-73]. The degree of formalization limits the possible content of the model, i.e. the more formalized a model is, the more rules it contains. As a result, flexibility is decreased, resulting in the omission of information in highly formalized models [compare FUCHS 2004, p. 76]. The degree of concretization strongly connects to the entities depicted. Whereas functional elements, whether unwanted or desired, are limited regarding their degree of concretization, an existing physical object, e.g. a product’s part, can be described on various levels of concretization. The level of detail of a model, on the other hand, is not limited by the described entities, as the different domains can be described on any desired level of detail, separating the degree of concretization strongly from that of detail. With the degree of concretization, it is the overall system and the system boundaries, for example the product architecture, which establish the limits. The level of detail, on the other hand, can be defined for each type of entity individually.

The above section discussed in detail the product architecture requirements comprehensive approach to develop a model. Fundamental modeling types were discussed concerning their adequacy. It was pointed out that schematic or descriptive modeling techniques are the most promising for fulfilling the requirements. The following chapter will detail the results and transfer them into a model for product architecture management, which will be outlined in the results of chapter 5.

4.3 Product architecture model and framework – outline

Subsequent to the discussion of the product architecture itself and the possibilities of modeling systems, this chapter will combine the results of both discussions into a cogent overview of the management of product architectures. To cope with the framework and to detail its content, chapters 5, 6, and 7 will clarify both the usage of methods and a procedural model on how to practically apply the framework. At this point, the product architecture framework consists of a general modeling approach, relevant categories or domains of entities, and a structuring of the product architecture according to the superior differentiation into goal-, object- and action-system. The framework will be refined in the following chapter, based on existing methods incorporated into the overall approach.



	1	2	3	4	5	6	7	8	9	10	11
1	■								☒		
2		■								☒	
3			■		☒						☒
4				■	☒	☒				☒	
5			☒	☒	■	☒					
6				☒	☒	■	☒				
7						☒	■	☒			
8							☒	■			
9	☒		☒						■		
10		☒		☒						■	
11			☒								■

Figure 4-1 Modeling approach based on matrix and graph representation

For the modeling of product architectures, the graphical and tabular modeling, i.e. graph and matrix representation, were chosen because existing models in systems engineering (discussed in chapter 3.2 and more thoroughly addressed in chapter 5) support these modeling techniques of the physical architecture, as well as other aspects of the product architecture, thus fulfilling the premise of an integrating comprehensive approach; this was discussed in further detail in chapter 4.2. In addition, the communication among stakeholders as an integral aspect of the process of systems architecting has to be supported by models easily understood by different professions and able to depict their different perspectives. The chosen modeling approach is both generic and closely related to the models used in the respective disciplines, thus enabling the interconnection of disciplines on the one hand, and the subsequent processing of the models within the respective disciplines.

MAURER sums up the benefits of the chosen modeling approach as follows [MAURER 2007, pp. 109-110]: techniques based on matrices are indispensable, especially for comprehensive analysis approaches. Systematic information acquisition is enabled through matrices; matrix representation is then applied and shared by the involved disciplines. Graph representation as a complementary model compensates for the shortcomings of matrix-based techniques, as graphs can be grasped rather intuitively, and the models can be transformed into one another. BONGULIELMI ET AL. support the importance of tabular or matrix-based approaches by giving numerous established examples, which are also discussed later in this work [BONGULIELMI et al. 2002].

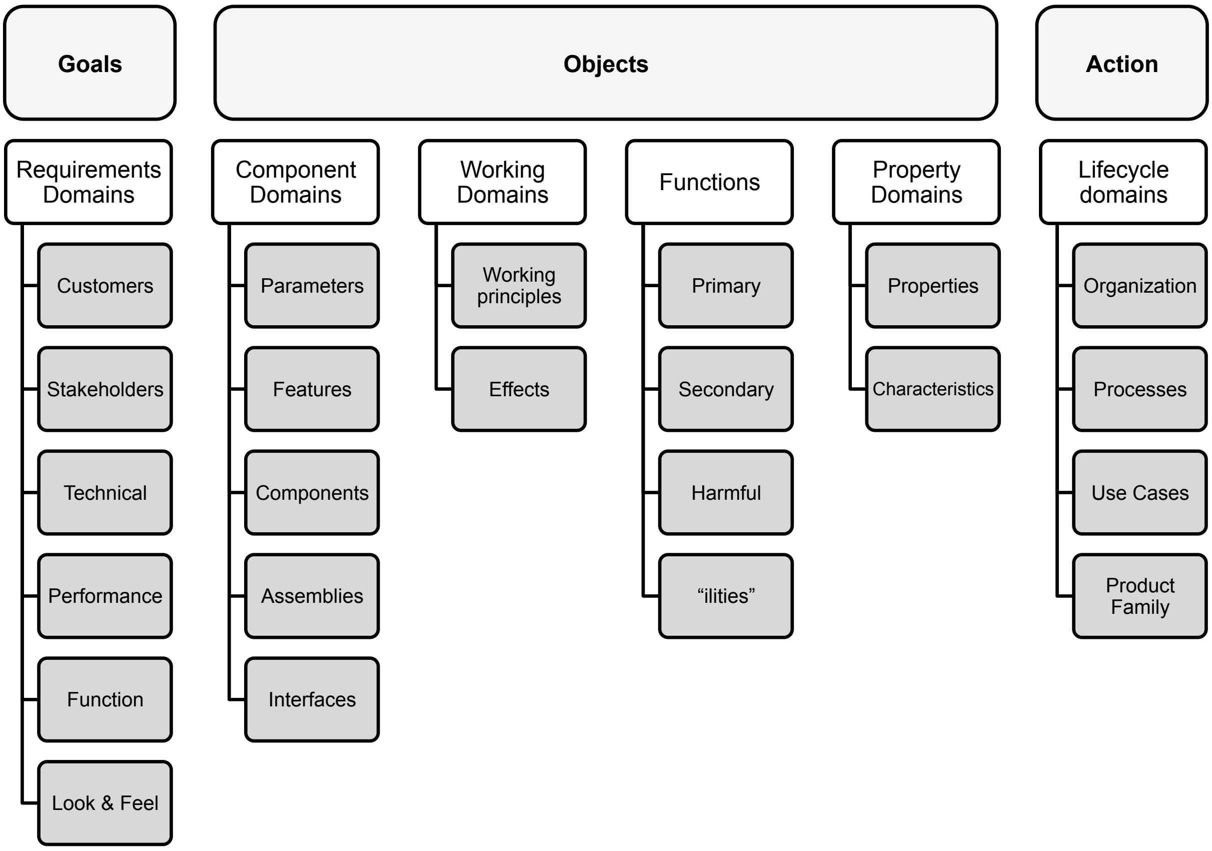


Figure 4-2 Domains of the product architecture model (framework)

The preceding figure depicts the domains of the product architecture model. The entities of the product architecture are grouped into the following domains: requirements, (physical) components, working principles, functions, properties, and organizational matters, which are addressed within the course of the product lifecycle. The domains can again be clustered into the system of goals, the system of objects and the system of actions. For the model presented, a hierarchy within the domains was deliberately avoided. The resulting entities within the domains represent distinct classes, separated from one another due to clearly definable differences. The following three sections discuss these groups and entities in detail, as well as the differences between them. The focus of the presented work is the system of objects, which cannot sufficiently be discussed and managed without the knowledge about the systems of goals and action. The dependencies between the different product architecture entities are addressed in a general manner at this point. The refinement of the product architecture will be based on a comprehensive literature review in chapter 5.

4.3.1 System of goals

The system of goals is composed solely of the domain of requirements. Within the domain of requirements, six different classes exist, grouping requirements with distinct characteristics and resulting in a reasonable classification.

Customer requirements stand for the voice of the customer, i.e. the company-external buyer and often also the user of the product.[55] Requirements expressed by customers can often be described as qualitative and incomplete. As such, customer requirements usually address performance and functional requirements or the look and feel of the product, rather than precise technical requirements. It is the challenge of the systems architect to translate these vague requirements into precise technical requirements, which a designer can again translate into desired properties and characteristics realized by physical components fulfilling the requirements.

The **requirements expressed by other stakeholders**, or stakeholders in general, result in technical, performance or functional requirements. Depending on the stakeholder in question, the requirements are expressed qualitatively or quantitatively. Stakeholders from company-internal departments, as well as external stakeholders, may chiefly address "ilities", qualitative measures of manufacturability or recyclability, for example, rather than primary or secondary functions.

Technical requirements are proposed by various sources of the goal- and action-system. Use cases or the product family imply certain types of energy used, for example excluding nuclear powered products in most cases, and pose other technical implications. Vaguely expressed requirements by stakeholders result in precise technical requirements, once the technical solution is more specified. Technical requirements themselves are quantitatively describable and thus, in most cases, include a property and a desired value of that property, such as a required length

[55] Often, the buyer of a product is not the user. For example, the buyer of a truck is usually the trucking company, while the user is the truck driver. The responsibility for maintenance might be with a third party etc. The acknowledgement of these circumstances is important when classifying, analyzing and assessing requirements and can be accessed by measures discussed in chapter 5.3.1.

Performance requirements specify certain functions of the product, giving specifications on how well the product fulfills a function. If acceleration is a desired function, the performance requirement specifies the characteristic of that function quantitatively, i.e. in what period of time a certain speed is to be reached.

The **functional requirement,** on the other hand, defines that acceleration per se has to be possible. Functional requirements usually stem from customers or other stakeholders, but are proposed by entities of the action system as well. The functions desired in functional requirements might express primary or secondary requirements and "ilities" as well.

4.3.2 System of objects

The system of objects represents the levels of product concretization by encompassing functions, working principles, and components of the product. The properties of the resulting solution or possible alternatives complete the picture of the product representation throughout the process of concretization.

Parameters stand for quantifiable measures, describing in detail the features, components, assemblies or interfaces of the product architecture. They stem directly from desired properties or characteristics of the product architecture and represent the smallest entity of the component architecture. Parameters apply not only for mechanical entities, but also for e.g. electronic (power, voltage etc.) or service (time for delivery etc.) components.

Features, a term stemming largely from computer aided design, represent a combination of parameters. Features are often standardized and required for interfaces or enabling the use of bought-in parts.

Components are the smallest inseparable unit of component architecture. The main characteristic of components, in contrast to parameters or features, is that they are the smallest entity that can independently provide a function of the product. They are defined through parameters and features and inherit that defined combination. The defined combination of features enables the adaptation of components to new requirements through an adaptation of parameters and/or features, resulting for example in altered characteristics or properties of the product. Within a product family, the same component can have different parameters and/or features. This allows for the scalability of products within the family or the realization of functions in individual products of the family that are not available to all products of the same product family.

The existence of **interfaces** is regularly considered as the mere physical (or energetic, material, geometrical, functional etc.) coupling of components (or aggregations, such as assemblies thereof). In the context of this work, interfaces are considered to explicitly support their definition and preservation along the process and within product families. In highly complex and differentiated product families especially, the explicit consideration of interfaces supports the keeping track of the defined interfaces.

Effects and working principles are inherited by the layout of components and apply to physical components exclusively, though the transfer of the abstract idea to different areas is not impossible. In that context, effects describe known physical or chemical behaviors, which in combination can be practically applied as working principles; these again usually represent a combination of a number of effects.

Functional entities can be differentiated mainly by their purpose, i.e. primary, secondary or harmful functions and “ilities”. **Primary functions** represent the main purpose of the product architecture, while **secondary functions** are required for the product to be able to fulfill the primary function. **Harmful functions**, though undesired and mostly harmful, result from the choice of useful functions, which, in combination, cannot be carried out without side-effects; these again are often compensated for through the use of secondary functions, e.g. the provision of a cooling system for processor chips. “**Ilities**” can be described as functions of a third degree, which are neither necessary for the fulfillment of primary or secondary functions, but are necessary from a lifecycle perspective. Typical known “ilities” are the different existing Design for X aspects, which were already discussed, including manufacturability etc.

The architecture properties, as the last domain of the object system, are divided into **characteristics** and **properties**. Properties then result from characteristics, which can be directly influenced by the designer through the variation of parameters and features. Properties correspond with the requirements of function and behavior, while characteristics correspond largely with technical requirements.

4.3.3 System of action

The system of action as the final part of the architecture model represents the product architecture entities relevant to the product lifecycle. Organization and process, use cases, as well as the product family, are all parts of the system of action. The entities of the system of action mainly represent sources of requirements of the product, providing boundary conditions and a basis for decision-making processes.

The **organizational entities** of the product architecture encompass not only the organizational situation within the company, but state-specific entities relevant for successful product development. These entities contain organizational and cost data and further aspects, which translate into requirements, properties etc., depending on the considered case. These entities are given relevance in decision-making processes, where economical and strategic decisions are necessary. The organizational entities will be discussed further in the following chapters, and a highly relevant to the means of variant management and the lifecycle perspective taken in chapter 5.7.

The **process domain** of the product architecture model represents the documented decision-making processes during product architecture management. As such, the process domain enables the linkage of the product architecture to process improvement measures and the replicability of the decisions and steps taken during product architecture management.

Use cases are generated to identify requirements stemming from scenarios in which user interaction with the product architecture is depicted. Though the term “user” is usually associated with the end-user of the product’s functions, use cases encompass all stakeholders within or outside of the company who interact with the product in any way during its lifecycle including design, production, delivery, use, recycling etc. Use cases and scenarios are highly relevant sources for functional requirements, for the identification of harmful functions or potential for innovations.

The **product family** is relevant as an entity of the product architecture model because it poses a number of boundary conditions, potentials and requirements on the product architecture. A distinct domain of entities was chosen to complete the picture of the product architecture and its role in the manufacturing industry today. Examples above include the sub-ordinate domains of product lines and product types.

5. Coping with product architecture

Following the overview in the previous chapter of the character, properties and constituent parts of the product architecture, the following paragraphs discuss existing methods, addressing one or more aspects of the product architecture. To allow for a coherent overview, a structuring of the later presented methods is discussed in the first chapter, underlining once again that there are many facets to product architecture which can be viewed from different perspectives. The following chapters discuss methods, approaches and theories, their goals, procedures and models, focusing on the role of the product architecture and the considered entities and their interrelations. The goal of this chapter is the refinement of the requirements to a comprehensive approach on product architecture management and the completion of the product architecture framework by identifying and specifying the product architecture entities and interrelations of entities.

5.1 Structuring the state of the art

Different possibilities exist to approach the variety of methods, approaches and theories in product architecture management. ZANKER enumerates the different possibilities to structure a number of methods based on a literature review: namely, the steps of the problem solving cycle; steps or phases of the design process; applicability (generic or specialized); considered system (organization, product etc.); integratability and special criteria [ZANKER 1999, p. 44]. Some of these options were discussed in the previous chapters. The motivation for the intensive discussion of product architectures was laid out, in accordance with the origins of complexity when coping with product architectures, i.e. the markets, organizations and organizational surroundings, and the inherent processes. A structuring in accordance with that classification does not enable a differentiation of phases of the process, yet offers a structuring of what is in the focus under consideration. To differentiate between certain phases of the process, a structuring in accordance with the development process appears to be feasible, yet neglects the iterative and recursive character of the process and suggests a continuous concretization of the product architecture, which is rarely the case. As this work focuses on the early phases of product development, a structuring according to the product lifecycle turns out to be unreasonable, as a reduction of the focus was already conducted.

As a compromise, the outlines of a problem solving process were chosen, allowing for the differentiation of tasks to be conducted, such as analysis, synthesis or decision-making, but recognizing the iterative and recursive character. As a rough outline, the differentiation into goal-, object- and action-system was chosen, with each subject detailed into different topics with the respective state of the art. It is clearly not possible and equally undesirable to precisely disconnect the three areas from one another. The system of action poses requirements to be considered in the system of goals, while different limitations of the object system might affect the system of goals etc. Nevertheless, to give an overview of the state of the art of existing methods, the differentiation of these three areas seems to be feasible.

In detail, the three pillars of the state of the art are structured as follows. The system of goals is divided into the gathering of requirements and the management of requirements along the process, while the action system covers the support through computer-assisted means as well as the downstream-activities, i.e. the consideration of the whole lifecycle, throughout which the issue of variant management and product families is considered explicitly. The object system representing the product architecture, and covering different means of system analysis, synthesis and evaluation is the most importance for the discussion in this work. The structure of the state of the art is summed up in following figure.

The result of the following discussions will be the analysis of the extent to which the overall requirements, discussed in chapter 1.2, are met by the existing approaches, and which requirements remain and can be detailed. The means that are still necessary will then be developed in chapters 6 and 7, while chapter 8 proposes a validation example and the final discussion of results.

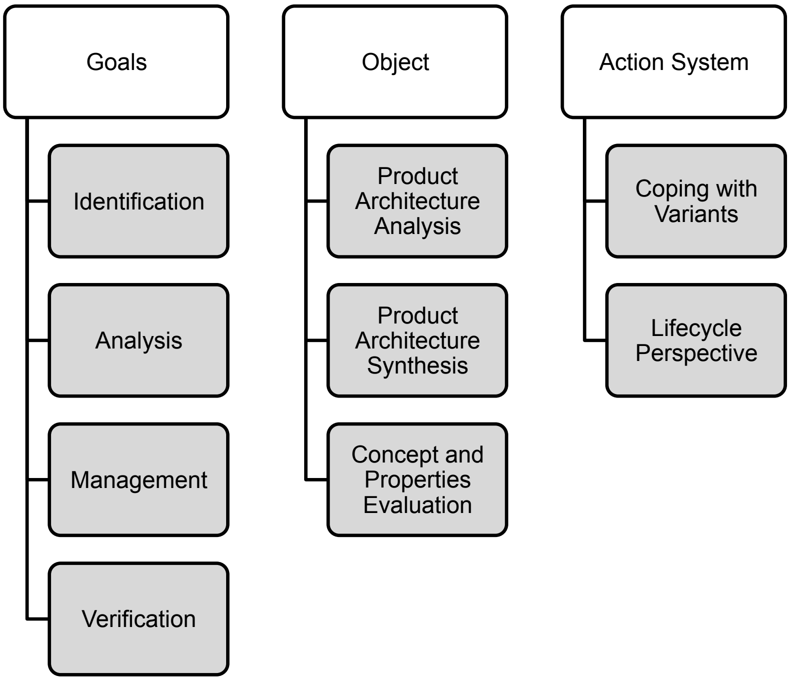


Figure 5-1 Structuring the state of the art in product architecture management

As was stated in the definition of requirements in chapter 1.2, it is not the goal of the following discussions to point out the shortcomings of different approaches or to isolate the approach that this work seeks from existing approaches. On the contrary, the presented approaches are considered beneficial, at least in combination with one another, and thus constitute the fragments of a comprehensive approach, which the presented work aims to unite.

Two approaches in systems engineering were mentioned in chapter 3.2.4, which aim for a comprehensive approach for managing product architectures as well, namely the different Architecture Frameworks (AF) and the Systems Modeling Language (SysML). The following chapter will briefly introduce these two approaches and set them into context for the following work.

5.2 Comprehensive approaches in systems engineering

Before the following chapters enter into detail regarding the system of goals, objects and the action system, this chapter introduces two approaches that discuss architectures from similar points of view. In particular, these approaches are the architecture frameworks and the Systems Modeling Language (SysML). While architecture frameworks propose a methodological outline for coping with architectures (mainly of information systems and enterprises, yet with adaptable outcomes for product architectures discussed in this work), the SysML states and approach for the modeling of entities especially relevant in the context of systems engineering. Following the discussion of these two approaches, the upcoming chapters will discuss the state of the art in detail, according to the system of goals, objects and the action system.

5.2.1 Architecture frameworks

Briefly introduced in chapter 3.2.4, this chapter will discuss architecture frameworks more comprehensively. Architecture frameworks can be differentiated into architecture frameworks for **product architectures** [e.g. ZACHMAN 1987, DOD 1997], **enterprise architectures** (organizations) [SANTE et al. 2007, MATTHES 2011], and, as proposed more recently, **process architectures** [BROWNING 2009]. **Integrated architecture frameworks** aim for the combination of not only different views of one type of architecture, but also strive to combine different architectures, such as process, enterprise and product architecture [KRUCHTEN 1995, WOUT et al. 2010]. The following paragraphs will point out the common ground on which these architecture frameworks define their principles.[56] Furthermore, differences and commonalities between the approach presented in this work and the architecture frameworks will be pointed out. To anticipate the results of this chapter, the philosophy of architecture frameworks underlines the importance and goals of this work as discussed in the chapter 1.2, chapter 4, and chapter 5.8. Yet, in the context of engineering design research, a number of needs remain unanswered, and are addressed in this work.

Both this work and the architecture frameworks aim for the **coping with complex systems** [BROWNING 2009]. Complex systems considered in architecture frameworks are large technical systems, products in general, similar to systems engineering developed in the context of military applications [DOD 1997, MOD 2005], or information systems and enterprises [SANTE et al. 2007, MATTHES 2011]. The consideration of enterprise architectures

[56] Popular and representative existing frameworks are e.g. The Open Group Architecture Framework (TOGAF) [SANTE et al. 2007], the Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance Architecture Framework (C4ISR or DoDAF) [DOD 1997], the Integrated Architecture Framework (IAF) [WOUT et al. 2010], or the Ministry of Defense Architecture Framework (MoDAF) [MOD 2005].

is often very strongly related to the information system background [compare MATTHES 2011], i.e. the enterprise is perceived as a system to be developed with information systems and to be provided with support from the information systems. Depending on the architecture framework in question, the enterprise may be considered as the combination of e.g. people, processes, physical structures, and engineering and information systems [MOD 2005, p. 10].

This also shows in the characterization of rather comprehensive architecture frameworks. WOUT et al. for example put forward an Integrated Architecture Framework (IAF), with a focus on comprehensiveness. The proposed IAF considers business architecture, information architecture, information systems architecture, and technology infrastructure architecture [WOUT et al. 2010, pp. 53 ff.], underlining the close relationship to information systems. While information systems were the origin of architecture frameworks [see ZACHMAN 1987], the **principles are considered to be adaptable** to “physical products” or other types of products as well [BROWNING 2009].

Similar to the need identified in chapter 4.2, the purpose of architecture frameworks in general is to **structure the different views of a complex system** [BROWNING 2009]. Architecture frameworks intend to give an overview of complex systems, and thus reduce the perceived complexity through the combination of models from different viewpoints [SANTE et al. 2007]. Thereby, each view is comprised of a number of system entities and their attributes, a guideline and semantics (a common, defined vocabulary) for their representation [BROWNING 2009].

Another similarity shows in the characterization of development as a decision-making process.[57] Based on that observation, an architecture framework is designed to **support the decision-making process** by providing relevant information in a common description, and tying together the different views of the architecture [compare e.g. BROWNING 2009, SANTE et al. 2007, p. 9]. Similar to the goals of this work, a common model that unites the existing views into an underlying models and semantics is considered a key requirement for a successful coping with architectures [MOD 2005, p. 11].

Having established the groundwork of architecture frameworks, the following paragraphs discuss a number of key ideas and principles of architecture frameworks, which integrate into the presented work and accompanying challenges. The presented work does not then aspire to compete with existing architecture frameworks. As the above discussion has shown, there are concepts and ideas in both areas worth discussing and coupling to the approach presented in this work.

The following paragraphs will take a closer look at the presented viewpoints and combined within the different architecture frameworks. Further discussions will include the models used within the architecture frameworks. The conclusion of this chapter will discuss further characteristics of architecture frameworks, the similarities and difference of the enterprise-driven architecture frameworks and the approach presented in this work.

[57] Compare the discussion of chapter 1.1.3 and chapter 5.6.

Each framework tries to define a set of "model categories", i.e. groups of previously mentioned views or single views [MoD 2005, p. 10]. Within the Ministry of Defense Architecture Framework (MoDAF), six viewpoints are established to categorize 38 different views. These six viewpoints include a differentiation according to strategic (management and planning perspective), systems (functionality, interconnectivity, etc.), technical (standards, constraints, etc.), operational (operational processes, operational analysis, developed requirements, etc.), and acquisition viewpoints (program dependencies, timelines, etc.). A sixth viewpoint is defined for "all views", which inherits summary information to be indexed and verified [MoD 2005, p. 14]. In a similar approach, the "4+1" view model defines the name, giving four views, plus one rather comprehensive view. The four views of the "4+1" view model include the logical view (end-user functionality), development view (programmers, software management), physical view (system engineers, topology, communications), and the process view (integrators, performance, scalability). More global entities, such as scenarios or use cases to illustrate the system, are grouped and understood under the "+1" view [KRUCHTEN 1995, p. 2]. The C4ISR or DoDAF differentiate between three distinct categories of views: operational (tasks, activities, information flows), systems (descriptions and graphical representations of systems, i.e. physical representation, meta-data, performance parameters and operational requirements), and technical views (standards and conventions, i.e. the rules to ensure that a system fulfills its requirements) [DOD 1997, p. 2-1 ff.]. The Integrated Architecture Framework defines its views (or levels) on a rather abstract scale. The differentiation of the levels of the IAF is provided by the leading questions to be answered on each level. Four levels are considered in the framework, namely the levels answering the questions "Why?" (contextual), "What?" (conceptual), "How?" (logical) and "With what?" (physical). The question of "When?" is explicitly left out of the framework, yet is mentioned and not considered less important [WOUT et al. 2010, p. 237].

The discussed architecture frameworks provide a comprehensive enterprise-driven view of the architecture. While operational demands and requirements, use cases, processes etc. play an important role in architecture frameworks, due to the enterprise context, the interplay with those entities might vary for the purpose of the method proposed in this work. Nevertheless, the discussed architecture frameworks give an overview of the numerous classes of the architecture entities, and reflect not only in the structuring of this chapter 5, but also in the chosen model and its entities in the approach presented in this work.[58]

Each architectural view contains implications, stating which entities of the architecture are to be displayed within the view. These views partially overlap, resulting in the description of the same entities in different views [DOD 1997, p. 2-1]. In addition, the views are partly redundant and integrated views are beneficial. Within the C4ISR approach, similar to the "4+1" view model, it is stated that integrated views, which combine multiple single views, provide an advantage compared with the models of single views. [DOD 1997, p. 2-1 f., KRUCHTEN 1995, p. 2]. For the approach presented in this work, the discussion focuses on domains, representing classes of architecture entities. The groups of views and single views, as presented in architecture frameworks, are implicitly available through the combination of

[58] Compare chapter 4 and chapter 7.2.

domains and their interrelations within one view. As will be discussed in chapter 7, the required views can be derived from the problem or project at hand, as well as from the perspective of organizational units and responsibilities within the enterprise. Claiming to be applicable in a generic fashion, the presented approach does not deliberately predefine views.

The approaches concerning the models used within architecture frameworks differ. While within the MoDAF, the UML is the main modeling method used for the object oriented modeling of systems [MoD 2005, p. 23], other frameworks propose different approaches. The "4+1" view framework, for example, provides distinct notations for each view [Kruchten 1995, pp. 2 ff.]. Within the C4ISR, the information on how architectures are to be modeled is defined in the C4ISR Architecture Data Model (CADM), a meta-model or logical schema in the form of an entity-relationship diagram [DoD 1997, pp. 4-87 ff.]. Defined notations exist for the documentation of the specific problem domains and features of the architecture, such as system interface descriptions, system evolution diagrams, systems functionality descriptions, etc. [DoD 1997, pp. 4-1 ff.].

It is clear that there is a need for a defined notation for the presented architecture frameworks, each designed for a specific enterprise. For the purpose of a modeling approach for the presented work, a discussion of requirements of architecture modeling was conducted in chapter 4.2. Since the approach of this work aims for a largely generic approach, with the possibility to expand and complete the model and proceeding at will. Furthermore, the goal is to be able to integrate a number of existing models and methods in an integrated approach. For that purpose, the entities of the architecture are identified and modeled in an entity-relationship diagram. The generic matrix representation was chosen for the modeling, enabling the documentation of information of UML-diagrams, as well as different other notations.[59]

The C4ISR framework includes not only the modeling approach and views for architectures, but also proposes a six-step process of building an architecture, the "architecture description process", as well. The steps include the determination of the use of the architecture (1), determination of the scope of the architecture (2), determination of the characteristics to be captured (3), determination of the views and products to be built (4), building of the requisite products (5), and the use of the architecture for the intended purpose (6) [DoD 1997, p. 3-5]. The process documented here points out that architecture development is often considered separate from the design of products. Architecture frameworks aim for the phase before design, separating architecture design from detail design [compare Simmons 2008, p. 18]. In this work, the goal is to equally support the architecture development process, while integrating its outcome with detail design and respecting the iterative interplay of the two processes.[60]

To sum up, a number of beneficial impulses and commonalities show in the discussion of architecture frameworks in the context of this work. First of all, the acknowledgement of multiple views, and the importance of their considerations, is important. The interrelations

[59] See chapter 6.1 and 6.2 for examples and the general possibilities.

[60] The iterative character of the design process was intensively discussed in chapter 1.1.3.

between views are agreed upon, as well as the overlapping of different views and the accompanying complexity of architectures (considered also as redundancy [DoD 1997, p. 2-1]). All sources agree on the necessity of a comprehensive view of the architecture, aside from the single views.

Still, two major differences show: the modeling approaches differ, although entity-relationship diagrams prevail and are a similar approach, compared with the model chosen in this work. The views are not predefined within the MDM approach, or within the approach outlined in this work. While architecture frameworks provide predefined views necessary for the considered enterprises, a generic application is not enabled, due to restricted possibilities. As discussed earlier, the domain-based meta-model of the presented approach includes implicit views, which can be explicitly defined if necessary. Valuable input comes from the discussion of architecture frameworks, supporting the ideas, goals, and general direction of the presented work.

5.2.2 Systems Modeling Language (SysML)

The Systems Modeling Language was developed as a modeling language to support systems engineering. It enhanced the Unified Modeling Language by several items necessary for systems engineering, and omits items unnecessary. UML itself was developed primarily as a standard for software engineering [Holt & Perry 2008, p. 23, Weilkiens 2008, p. 16]. A prominent example of a UML enhancement is the lack of requirements modeling, for example [Holt & Perry 2008, p. 27, Weilkiens 2008, p. 16]. The following paragraphs will briefly discuss the characteristics of SysML and its implications for the presented work.

SysML seeks to enable system engineers to capture and model system requirements, system behavior and the system structure, while e requirements and behavior in particular required the extension of UML [Holt & Perry 2008, p. 27]. A system model in SysML is comprised of three interrelated models of the **system structure** (block definition diagram, internal block diagram, parametric diagram, package diagram), **system behavior** (activity diagram, sequence diagram, state machine diagram, use case diagram), and **requirements** [Friedenthal et al. 2009, p. 30, Hause 2006, Wölkl & Shea 2009]. Parametrics are often considered to be the fourth pillar of SysML [Friedenthal et al. 2009, p. 18, Hause 2006]. The system model is related to engineering analysis and simulation models [Friedenthal et al. 2009, p. 18, Johnson et al. 2007].[61]

In the context of this work, especially the early phases of systems architecting are of interest. Several authors embedded the SysML approach into a procedural model and analyzed the capability of the language to model the process-results in the early phases. SysML itself as a modeling language does not claim to solve the procedural aspects of designing architectures within the approach.

[61] For a detailed description of SysML language and syntax see for example the documentation of Friedenthal et al. [Friedenthal et al. 2009, pp. 63 ff.].

WÖLKL & SHEA choose a computational approach and provide a practical example from the automotive sector. The findings support the general applicability of SysML, pointing out possibilities to conveniently relate different models and entities of the architecture, based on the possibility of the computer-aided management of architecture entities [WÖLKL & SHEA 2009]. In a similar manner, GANESAN & PREVOSTINI aim for the depiction of the Design Solution Space in SysML, i.e. the mathematical documentation of multiple alternatives for a given design problem [GANESAN & PREVOSTINI 2006]. The analysis of solutions and the identification of the most suitable solution in particular turn out to be the major challenge when designing, and thus are reflected in the application of SysML. Other works focus equally on the early phase of design, integrating established methods and approaches into SysML. TURKI & SORIANO successfully define an extension of SysML for the depiction of Bond-graphs in activity diagrams, using the standard extensions available [TURKI & SORIANO 2005].

The approaches discussed above can be generalized, stating that SysML enables the integration and/or coupling of different views of the same problem.[62] SHAH et al. are trying to assess precisely this capability by using the example of a simple mechatronic system and domain-specific representations of it [SHAH et al. 2009]. Again, the authors reach the same conclusion, stating that the possible mappings of views, as well as the general opportunities of SysML, support the processing of domain-specific knowledge and models [SHAH et al. 2009]. THRAMBOULIDIS provides a similar use case, stating that challenges remain, above all the barriers of integrating the different views of mechatronics, which can be extrapolated to the general challenge of interrelating different views of different disciplines [THRAMBOULIDIS 2010].

To conclude this brief chapter on SysML, the implications for the presented work can be summarized as follows. Similar to architecture frameworks, SysML provides reasonable input regarding which domains and entities of the architecture to consider when discussing the management of product architectures in systems engineering. SysML underlines the importance of the interrelation of models and views of the architecture as well. The entity-relationship character of SysML is similar to the modeling approach chosen in this work, yet more formalized.[63] Still, the intention of both approaches is the same, i.e. provide a sound and more or less comprehensive basis for systems engineering, with the possibility of enriching the basic model through specific needs arising from projects and use cases. Since both approaches deliver a generic modeling approach, advantages and disadvantages show in both cases. Depending on the level of abstraction required at each project phase, e.g. the early phase of design, a high level of formalism may not be desired and/or feasible. On the other hand, a common semantic ground supports the interaction between users and the exchange of standardized models. The integration/interrelation of different models is desired and possible in both approaches, with the approach chosen in this work striving for the integration of

[62] Compare the similarity of the integration of multiple views in the discussion of architecture frameworks in chapter 5.2.1.

[63] Compare chapter 4.2.

different levels of abstraction as well.[64] Yet, SysML does not provide the means of analysis on a generic level, restricting the approach solely to modeling. Since both approaches claim the ability to integrate different models and views, a coupling of SysML to models in MDM notation may be feasible for certain aspects and needs. Either way, both approaches open the door for the integration of different models and views: one is far more formalized than the other, and thus possesses the advantages and disadvantages stemming from a formalized approach.

5.3 Goals and requirements

The importance of the objectives and technical requirements for products was discussed extensively in the introductory chapters, in particular chapter 1.1.1. The discussion of goals and requirements in the following paragraphs is divided into four major fields, depicting the process of requirements engineering as elicitation, analysis, management, and verification [MALETZ 2008, p. 35, compare JIAO & CHEN 2006]. The first area covers the approaches to identify and gain requirements and affordances of customers, while the second area structures and interprets requirements. The third area discusses how the requirements are portrayed, monitored and controlled along the process of design and development. The final stage depicts the verification and acceptance of requirements in the final product. The distinction was made firstly to be able to depict how the starting point of design is reached through identification of requirements, and secondly how they can be considered and updated during the process of design (i.e. the acquisition and the analysis and synthesis of requirements according to [LIU et al. 2001]).[65]

A major issue in requirements management is, according to LIU et al., the large number of existing methods and techniques for requirements acquisition and management, which lack a systematic process and framework that integrates different approaches and ties them together [LIU et al. 2001]. JIAO & CHEN identify incomplete and imprecise requirements as a major issue, as well as the lack of homogeneity of requirements documentation in the sense of quality and semantics, as well as inconsistent requirements specifications [JIAO & CHEN 2006, see also LIU et al. 2001, MALETZ 2008, p. 36], for which LIU et al. identify the insufficient guidance of the acquisition process as a cause. An efficient communication with the customer and maintaining focus on the most significant requirements in terms of relevance for customer satisfaction are further issues worth mentioning [LIU et al. 2001]. The list can be complemented by problems arising due to numerous stakeholders, their availability and perspectives, as well as the conflicts of requirements [JIAO & CHEN 2006, MALETZ 2008, p. 36]. When considering complex product service systems, the different involved disciplines, their requirements models, methods and requirements management processes differ, thus causing further difficulties.

[64] Compare chapter 6 for chapters on model integration and abstraction.

[65] JIAO & CHEN give a recent overview on the research issues in customer requirements management based on comprehensive references in [JIAO & CHEN 2006].

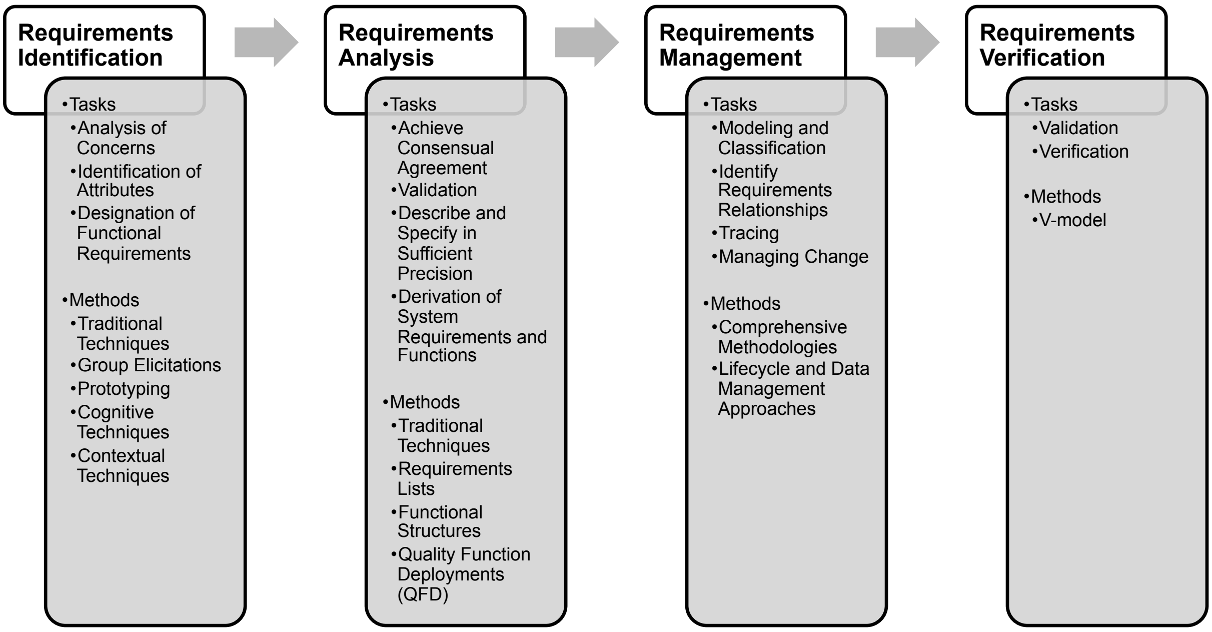

Requirements Identification
•Tasks
•Analysis of Concerns
•Identification of Attributes
•Designation of Functional Requirements
•Methods
•Traditional Techniques
•Group Elicitations
•Prototyping
•Cognitive Techniques
•Contextual Techniques
Requirements Analysis
•Tasks
•Achieve Consensual Agreement
•Validation
•Describe and Specify in Sufficient Precision
•Derivation of System Requirements and Functions
•Methods
•Traditional Techniques
•Requirements Lists
•Functional Structures
•Quality Function Deployments (QFD)
Requirements Management
•Tasks
•Modeling and Classification
•Identify Requirements Relationships
•Tracing
•Managing Change
•Methods
•Comprehensive Methodologies
•Lifecycle and Data Management Approaches
Requirements Verification
•Tasks
•Validation
•Verification
•Methods
•V-model

Figure 5-2 Requirements management process, respective tasks and methods (compare [MALETZ 2008, pp. 35ff.])

5.3.1 Identification of requirements

The goal of the first phase in requirements management is to make explicit the implicit customer verbatim constructs [JIAO & CHEN 2006]. The three major activities in this phase are the analysis of the concerns of different stakeholders, the identification of attributes and the identification of functional requirements [LIU et al. 2001].

The concerns of stakeholders are the drivers of requirements identification.[66] Especially important is the consideration of the differences between stakeholders, such as end users, maintainers, producers etc. Their integration into the process may be accomplished through traditional techniques such as group sessions, e.g. creativity sessions including different stakeholders, structured interviews with the stakeholders, questionnaires and surveys. The creation of use cases allows for the identification of user needs in cooperation with stakeholders or without them. The use case-based functional requirements acquisition allows for the identification of functional requirements from user viewpoints [LIU et al. 2001,

[66] In the context of concerns of stakeholders, the concept of affordance-based design is worth mentioning. An affordance structure is intended to encompass more than merely a product's functions, but its behavior and interaction with users and stakeholders as well. As such, different Design for X aspects are considered within the affordance structure, providing a broader view than functional structures [MAIER & FADEL 2001, see also MAIER & FADEL 2006]. The affordance structure thus aims at a product representation depicting product requirements based on different affordances (comparable to Design for X aspects).

MALETZ 2008, p. 37]. JIAO & CHEN point out numerous different ways of identifying customer requirements through different psychology-based means, applying artificial intelligence or methods of knowledge recovery in general [JIAO & CHEN 2006]. Concrete measures for a systematic requirements acquisition are listed by PONN & LINDEMANN, such as checklists and benchmarking [PONN & LINDEMANN 2008, p. 37]. KUSIAK introduces group meetings and interviews as information gathering methods [KUSIAK 1999, p. 10].

The identification of attributes relevant for different stakeholders is another task to support requirements identification. LIU et al. give the example of quality attributes, such as “number of defects”, from the viewpoint of the user or maintainer, of which “reliability” would be the superordinate concern [LIU et al. 2001]. DUHOVNIK et al. propose the use of tree diagrams to systematically decompose the concerns of stakeholders and obtain concrete attributes [DUHOVNIK et al. 2006].

As a third task, the definition of functional requirements is considered to be part of the requirements identification stage by LIU et al. [LIU et al. 2001] while e.g. MALETZ defines the derivation of functions as part of requirement analysis rather than elicitation [MALETZ 2008, p. 36]. PONN & LINDEMANN add the use of functional models or use case-oriented functional models for this purpose [PONN & LINDEMANN 2008, p. 40- 42].

5.3.2 Requirements analysis

The analysis of requirements should enable a prioritization and classification of requirements [JIAO & CHEN 2006]. Apart from these, the major issue after identifying and documenting requirements is the acknowledgement and systematic identification of requirements relationships, i.e. positive and/or negative influences among the numerous gathered requirements of the previous stage. In that context, LIU et al. concentrate on the analysis of pairs of gathered requirements, with the goal of eliminating redundant requirements. Requirements are redundant under three possible circumstances: if they are synonymous, i.e. representing the same subject yet formulated differently; if one requirement is formulated more generally, thus including more specific requirements; or if one requirement is stronger than another, thus including both requirements. LIU et al. describe these relationships as synonym, generalization, and strength [LIU et al. 2001].

The most critical relationships are those of opposing or contradictory nature. While LIU et al. state that the elimination of one of the contradictory requirements is the only option [LIU et al. 2001], other authors seek solutions fulfilling both requirements or reasonable tradeoffs to solve the conflict [JIAO & CHEN 2006, MALETZ 2008, p. 38, PONN & LINDEMANN 2008, p. 35].

PONN & LINDEMANN cite the Quality Function Deployment (QFD) approach for a systematic analysis of requirements and their relationships, in which the interrelations between customer requirements and technical properties (or characteristics) are gathered in an interrelation matrix. The identification of requirement conflicts is therefore depicted in a second, symmetrical, matrix of properties. These interrelations are then rated, whether the properties are (very) supportive or (very) contradictory [PONN & LINDEMANN 2008, pp. 41-43].[67] The approach, as proposed by PONN & LINDEMANN, appears to be more sufficient for a comprehensive analysis of requirements than approaches that merely consider requirements themselves. JUNG places similar importance on the relations of requirements, giving the example of a handheld vacuum cleaner. JUNG relates the system in the form of a decomposed component representation, different stakeholders or users, and a decomposed environment system with one another [JUNG 2006, p. 90]. Requirements are then derived by analysis of the resulting interrelations, the size of an identified flux of force between user and product for example [JUNG 2006, p. 95f.].

Since requirements are not dependent on one another per se, it is their indirect codependence that is based on the technical realization. As such, the comprehensive analysis allows for a systematic identification of contradictory requirements, and further enables the identification of technical characteristics and properties, which demand for innovative solutions to solve the conflicts.

5.3.3 Requirements management

The management of requirements, the monitoring and updating along the lifecycle are highly relevant for achieving successful and accepted products as a result [PONN & LINDEMANN 2008, p. 47, ILIE et al. 2008]; this is especially true for the development of software systems, where requirements change rapidly [O'NEAL 2003, pp. 8ff.], as well as in information-intensive environments such as the automotive industry [ILIE et al. 2008]. Different possibilities exist for the management of requirements along the product lifecycle, ranging from the administration of a dynamic requirements list [PONN & LINDEMANN 2008, pp. 47ff.] to different comprehensive approaches that are partly computer supported [for a brief discussion of different approaches see MALETZ 2008, pp. 42ff. and JIAO & CHEN 2006]. O'NEIL, for example, derives a graph-based mathematical approach for measuring the design impact of requirements changes in software development [O'NEAL 2003, pp. 45ff.]. KUSIAK describes a formalized approach for conceptual design, based on the decomposition of requirements and a formalized derivation of solutions; this approach uses a coupling of the functional solution space with the requirements space [KUSIAK 1999, pp. 201ff].

The difference between requirements in distinct disciplines and their management are evident, discussed for the areas of mechanical engineering, computer science and service engineering by BERKOVICH et al., resulting in a framework for requirements management and pointing out different approaches of the disciplines when coping with requirements [BERKOVICH et al. 2009, compare JUNG 2006, pp. 25-60].

As a foundation of requirements management, the following paragraphs discuss the classification of requirements, enabling a more differentiated view on requirements as a basis for requirements management. The classification of requirements supports the process of

[67] The Quality Function Deployment approach is described more comprehensively in chapter 5.6.3.

compiling, organizing, and analyzing the design during the design process [JIAO & CHEN 2006].

The classification of requirements, as for functions etc. can be conducted from different viewpoints [MALETZ 2008, p. 90]. DOVE uses the differentiation of proactive and reactive dynamics in the context of agile systems, for which the ability to react is of the highest importance, due to influences stemming from several “reality factors”. These reality factors include pace of technology, systems complexity, agile enterprise, globalization, human behavior, organizational behavior and threat sources for the area of security strategy requirements. More drivers for change, both short- and long-term, are described by STARK [STARK 2005, pp. 55ff.]. The classification of requirements in that context also includes requirements stemming from the needs of creation, improvement, migration or modification on the side of the proactive dynamics and the requirements resulting from correction, variation, expansion or contraction, and reconfiguration on the side of the reactive dynamics [DOVE 2006].

In general, different methods exist for the classification of requirements. JIAO & CHEN cite ontologies and taxonomies as means to systematically support the process, while PONN & LINDEMANN rely on structured lists for requirements management [JIAO & CHEN 2006, PONN & LINDEMANN 2008, pp. 47ff.]. MALETZ proposes a requirements classification system, giving a reasonable overview of requirements in engineering design. The requirements are classified in categories as product, organizational and process requirements. Product requirements are further divided into functional requirements, reflecting stakeholder concerns, and non-functional requirements, which directly correspond with product properties and characteristics, such as weight, height etc. [MALETZ 2008, pp. 91-92]. The requirements can be further classified by characteristics that apply to all previously mentioned classes of requirements. Requirements are thus differentiated into internal and external requirements, depending on whether the source of the requirement is company-internal or -external. Internal requirements can be derived from customer requirements, clarifying vague or unspecific customer requirements, for example [MALETZ 2008, p. 93]. Internal requirements might also stem from stakeholder concerns within the company, thus are not classified as external but are not derived from external customer requirements either. Further characteristics include whether requirements can be described qualitatively and quantitatively, and a hierarchical of requirements into primary, intermediate or final requirements [MALETZ 2008, p. 93].

All types of requirements represent properties of the product. The properties are defined by attributes and parameters. Attributes identify certain properties as meta- data, while parameters define the values and value ranges which describe properties for the particular case [MALETZ 2008, p. 93, compare LINDEMANN 2009].

5.3.4 Requirements verification and validation

The requirements verification and validation stage compares the design to the current state of the requirements [MALETZ 2008, p. 39]. The guideline VDI 2206 defines verification as the assurance of properties, and underlines that the validation and verification has to be conducted continually throughout the process, reflecting the developing design as well as the continuously evolving requirements against one another [VDI 2206, p. 30]. Verification ensures that the design meets the requirements, while the requirements to be verified are clearly measurable and the process of verification can be formalized. The validation is constructed around the concerns of stakeholders rather than the measurable requirements. Validation is therefore less formalized and reflects whether the requirements actually respond to the stakeholder concerns [VDI 2206, pp. 38-39]. Different methods come into question for verification and validation, depending on the kind of product under consideration. As an example from the area of mechatronic products, the guideline VDI 2206 proposes a number of possibilities, such as hardware or software in the loop (HIL and SIL) [VDI 2206, p. 41].

In literature, the verification and validation of products by measuring the meeting of requirements is not necessarily considered to be part of requirements management but rather as part of the overall process of design. For example, LINDEMANN considers the tasks "Properties Assessment" and "Ensuring Goal Achievement", of which the first can be considered the identification of measurable properties of the developed designs, while the second measures the final product against both the requirements and the stakeholder concerns [LINDEMANN 2009]. Methods for ensuring the achievement of goals are characterized by a prioritization of goals and/or requirements and the identification of effects if not achieving them. LINDEMANN enumerates negation as an approach to identify consequences if goals are not reached, a cause-and-effect analysis based on identified requirements, and a fault tree analysis. All aim for the identification of potential outcomes if the fulfillment of requirements fails, accompanied by a prioritization of requirements based on these considerations [LINDEMANN 2009, p. 184].

5.3.5 Conclusion

The above sections introduced the process for requirements management, including the discussion of the respective tasks and methods for each phase of the process. Based on the given sources and considerations, the following figure delivers a rough outline of classification requirements for the product architecture management approach.

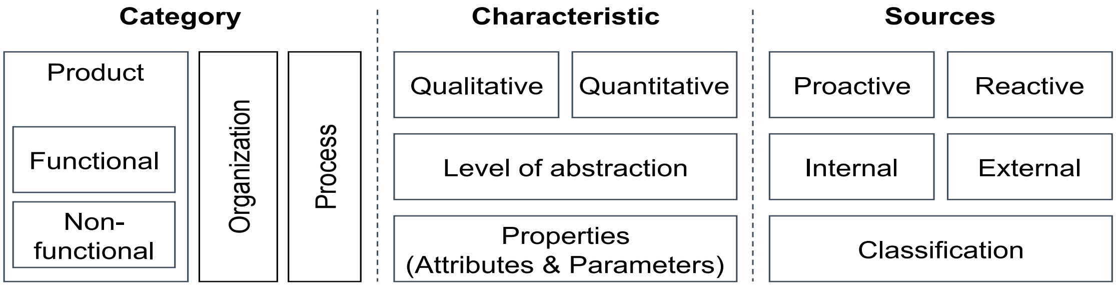


Figure 5-3 Requirements structure (compare [MALETZ 2008, pp. 91ff.])

Requirements in general may be categorized according to their target, i.e. product, organization or process. Requirements can be further characterized as qualitative or quantitative, address a defined level of abstraction of the product architecture (i.e. functions, working principles, geometrical solutions etc.), and define certain properties in terms of attributes and related parameters. The sources or stakeholders of requirements can originate from proactive or reactive behavior (for example reaction to markets or proactive development of markets), and might stem from company internal or external sources. Further classification of sources might differentiate according to requirements stemming from regulatory laws, safety and security, according to departments etc.

5.4 Product architecture analysis

Systems architecting is rarely the design of a system without predecessors, already existing subsystems or parts, similar existing systems or a product family into which the novel system has to be integrated. For that reason, the task of system analysis, or product architecture analysis in this case, is in any project a highly relevant step in the context of product architecture management and synthesis [PAHL et al. 2007, p. 81].

System analysis in general is closely related to, or even understood as, the "management of complexity" from a psychological or sociological point of view. In these areas, it is primarily the application of general principles and procedures that is discussed. BECK gives an overview on a number of procedures [BECK 2004, p. 57], including some which are discussed in this work. The procedures mentioned range from networked thinking to morphological charts, brainstorming, mind mapping, stakeholder analysis etc. BECK structures them in a fashion similar to chapter 5 in this work. The mentioned approaches differ greatly in their general applicability to different phases, including rather general schools of thought, as well as the support of certain tasks or phases, such as decision-making. Selected approaches are discussed in detail by different authors in [FISCH & BECK 2004]. For the work presented here, the structure of chapter 5 provides a reasonable frame for the discussion, while approaches that appear to be more from general schools of thought, such as systems thinking, rather than applicable methods and procedures, were discussed in chapter 3.2.

In the following sections, product architecture analysis is considered from three distinct viewpoints. First is the process of system analysis, providing the framework for a methodical approach of the subject. Second, an overview of the core principles of system analysis is provided, taking into consideration the fields of software engineering and psychology. In the last section, the analysis of complex networked structures is discussed; according to the definition of the product architecture and the systems approach, this is the most promising approach to use in the present context. As was stated in the definition of analysis, the boundaries separating system analysis for gaining information from coping with complexity are not clearly defined, as the two fields show a strong overlap.

5.4.1 Process of product architecture analysis

The process of product architecture analysis in particular, or system analysis in general, can be decomposed into a number of immanently necessary steps for a successful analysis. Depending on both the superordinate methodology in which the system analysis is embedded and the other characterizing circumstances of the situation, the approaches proposed by different authors vary slightly. In general, system analysis is the process of gaining information about a system and coping with the inherent complexity [PAHL et al. 2007, p. 58, compare FELGEN 2007, p. 28]. The following sections attempt to discuss a number of approaches to system analysis, capturing the differences of the approaches and reasons for those differences. As a result, a procedure of system analysis is presented that is suitable for the product architecture management approach discussed in this work, encompassing the different eventualities credited in the different discussed approaches.

PAHL et al. provide a general approach to the analysis of engineering systems with the three pillars of problem analysis, structure analysis and weak spot analysis. PAHL et al. emphasize the relevance of the analysis of the problem in the beginning, including the separation of the relevant elements from the irrelevant, as well as the decomposition of the problem into more individual parts. Structure analysis represents what is referred to by other authors as the identification of interrelations of elements [see e.g. PIMMLER & EPPINGER 1994] and classes of elements, i.e. similarities or repetitive features [PAHL et al. 2007, p. 58]. The analysis of weak spots within the system should support not only the identification of flawed system elements and working principles and their improvement, but also lead to the definition of alternative solutions based on identified weak spots [PAHL et al. 2007, p. 58]. Causes for weak spots can be not only system flaws, but also disturbing factors, such as changes of material properties and others, which can be identified through appropriate means, such as fault tree analysis or functional structures etc. [PAHL et al. 2007, p. 521].

Three steps are identified by PIMMLER & EPPINGER to detail the structure analysis referred to by PAHL et al.: namely, the identification of elements, interrelations and chunks of elements based on the interrelations between them. Elements are thus clustered into chunks, if the number of logical interrelations between them exceeds the number of interrelations with elements outside the chunk [PIMMLER & EPPINGER 1994]. The approach presented by PIMMLER & EPPINGER is designed to provide a guideline on how to apply the Design Structure Matrix (DSM), which is further discussed in chapter 5.4.3, for the analysis of complex engineering systems with the goal of defining appropriate modules [see e.g. BROWNING 2001].

In his systems approach to product complexity, STEINMEIER bases his procedure for product analysis on the system perception, i.e. a system consists of elements, system boundaries, and relations between elements, and elements and the system environment. Derived from the definition of the product architecture, STEINMEIER adds functions and properties, resulting in a procedure of six steps: identifying system elements [compare with system decomposition as in PIMMLER & EPPINGER 1994]; system boundaries; internal relations; relations to the system environment; functions; and relevant properties [STEINMEIER 1999, p. 30].

MAURER gives credit to the fact that analysis and synthesis in projects appear to be inseparable. Though agreeing with the general system perception of STEINMEIER, MAURER additionally provides an approach designed for practical applicability, resulting in a procedure of the following five steps: definition of the system under consideration, information acquisition, modeling, the systematic (structural) analysis itself, and a discussion of practice, i.e. the transfer of analysis results to the problem in reality [FELGEN et al. 2005b, p. 69, STEINMEIER 1999, p. 30]. For a systematic analysis, MAURER provides a comprehensive group of measures, based on abstract structural considerations derived from Graph Theory and

Network Science [MAURER 2007, pp. 197ff.]. In doing so, the former means of structure analysis, which were mainly considered to be the identification of chunks [PIMMLER & EPPINGER 1994, referred to as "grouping" by FELGEN et al. 2005b] in the context of product architectures, could be largely enhanced and broadened the possibilities and applicability of structure analysis.

A comprehensive procedure of complex system analysis is proposed by FELGEN et al., based on the discussion of existing procedural models [FELGEN et al. 2005b, compare PAHL et al. 2007, p. 58 and PIMMLER & EPPINGER 1994]. The resulting procedure adds a few aspects to the previously discussed approaches. Besides the coupling of different models and methods, all of which are based on a common database containing product information, FELGEN et al. provide a comprehensive approach with the additional perspective of software engineering, in which the identification of objects and classes is of high relevance [FELGEN et al. 2005b]. FELGEN et al. transfer this perspective to engineering, broadening what other authors call the identification of elements and interrelations. Especially for more comprehensive approaches to managing complexity, the relevance of classes (or domains) becomes apparent in engineering as well [see MAURER 2007, pp. 71ff.]. As a last feature, FELGEN et al. rightly emphasize the relevance of the identification of objectives as first step of the procedure, a step which e.g. MAURER considers cleared before conducting system definition [FELGEN et al. 2005b, MAURER 2007, p. 69]. The resulting procedure comprises of the steps to identify objectives, objects and classes, modeling the system, grouping of system elements [compare PIMMLER & EPPINGER 1994], verification, and analysis [FELGEN et al. 2005b]. It is important to note that misunderstandings due to the chosen wording of the authors might occur. In the presented procedure, the step "grouping" is actually what is considered by MAURER as "structure analysis", while "analysis" in the wording of FELGEN et al. is considered as "discussion of practice" by MAURER [FELGEN et al. 2005b, MAURER 2007, p. 69].

In an effort to provide a comprehensive proceeding based on the procedures outlined above, the following figure combines different, equally relevant, aspects to give respect to all recurring tasks and necessary aspects of system analysis. The resulting procedure intentionally includes the identification of the situation and resulting objectives as starting steps, as well as verification and implementation as concluding and reflecting steps. These steps appear to be necessary, as well as useful for the intended goal of establishing an approach for the management of product architectures. System decomposition, i.e. the identification of relevant system entities to solve the problem, is integrated, as is the process of information acquisition, whose importance in practice can not be stressed enough. The modeling of the system, as well as the actual analysis, are discussed separately in two steps, allowing for the use of existing models and methods, based on coherent information about the system.

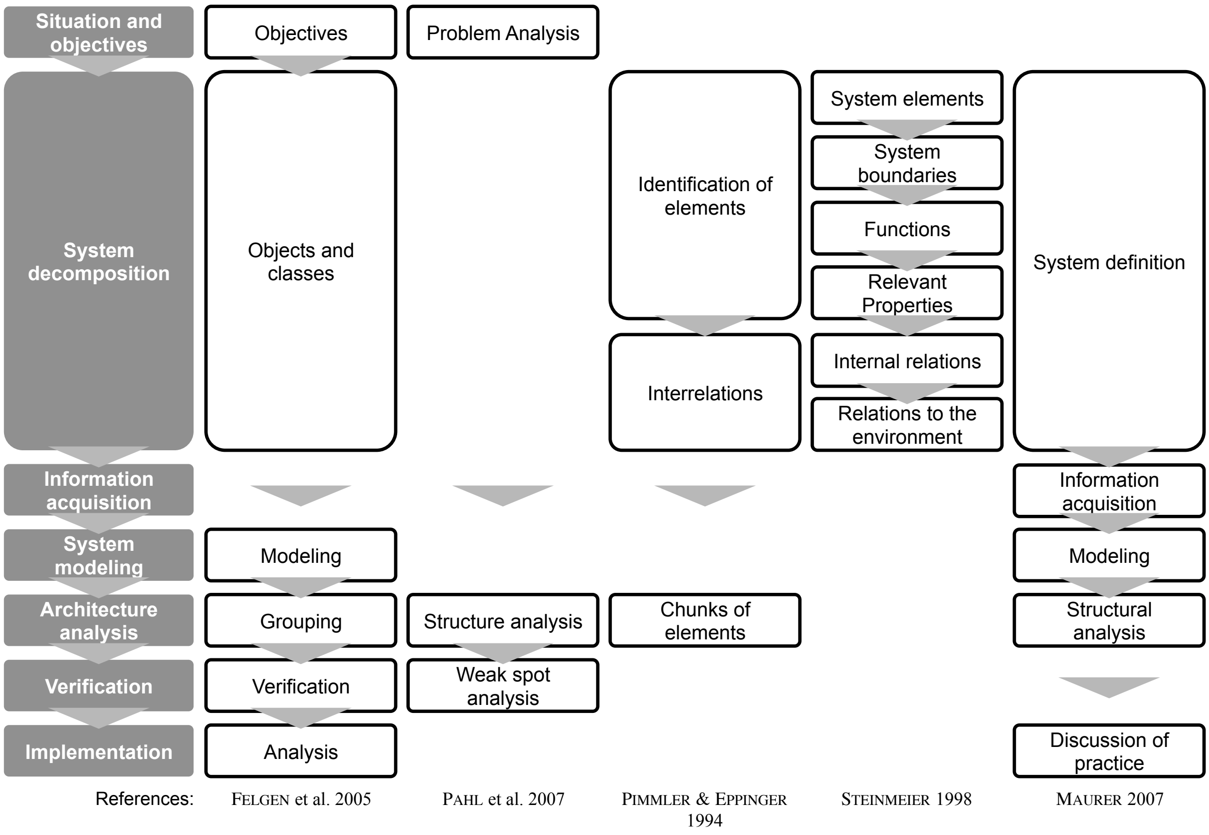


Figure 5-4 Process of system analysis (left) derived from different approaches

In the following sections, the steps of the procedure and possible applicable principles and methods are discussed. The methods and principles are not strictly related to certain steps of the procedure, as the distinct steps are usually overlapping, conducted iteratively or recursively, thus making a discrete differentiation of steps that is not only unnecessary, but also practically impossible. As the discussion of existing approaches for system analysis has shown, there are differences regarding how to conduct a reasonably comprehensive system analysis, depending on the goal and the method in use. The goals of this work, the requirements to the solution at the end of chapter 5 and the results of the discussion of the state of the art will point to which steps of the procedure are relevant. More importantly, this work will indicate which steps contribute to an approach, which is not only designed for system analysis, but also for the management of product architectures, including synthesis and portfolio maintenance.

5.4.2 Fundamental principles

Overviews of methods and principles to support solution finding processes show that proposed solutions range from complex methods and methodologies to fundamental principles, i.e. generically applicable logics [compare BECK 2004, p. 57]. The fundamental principles presented in the following sections can be differentiated into two major groups, depending on how the system analysis is approached. The first and important group of principles for system analysis is based on a change of perspective of the system under consideration. The second group of principles, originating largely from the area of software engineering, is the principles focusing on certain element properties, resulting in manageable groups of elements.[68]

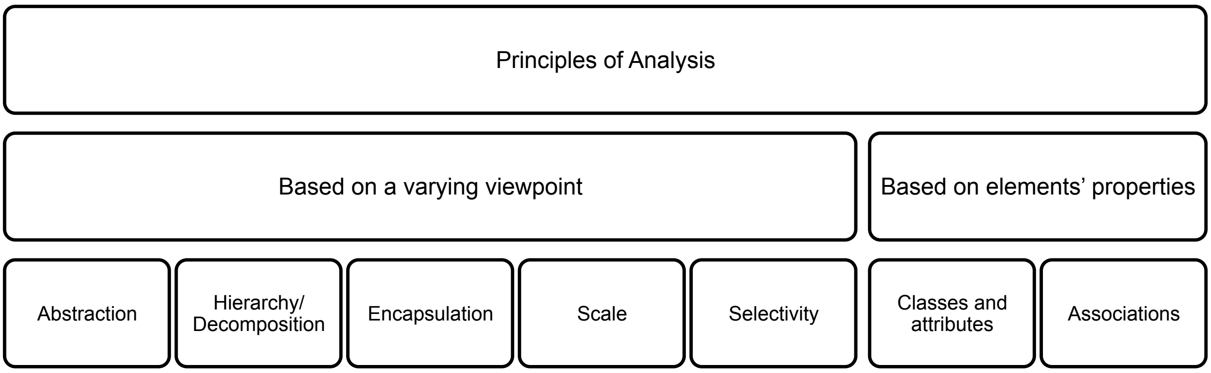


Figure 5-5 Fundamental principles of system analysis (compare F*ELGEN 2007, p. 17)*

It is important to note that the presented principles are rarely used discretely from one another. The typical method or approach uses a number of the principles of system analysis, as the following discussion will show. A model, for example, is a representation of an actual situation, which is both abstracted and usually out of scale, while the modeling process often includes a hierarchical decomposition of the system.

Abstraction

Abstraction as first principle of system analysis is, in general linguistic usage, the separation of the relevant from the irrelevant [LINDEMANN 2009]. BOOCH describes abstraction as the outside view of an object, separating function from realization [BOOCH 1994, pp. 60-63]. Descriptive models are regularly used as support of the first step of system analysis, being the identification of situation and objectives and the system decomposition. From the very beginning, abstraction is thus the fundamental principle accompanying the analysis process.

As in modeling in general, the identification of relevant and irrelevant system properties and elements is intrinsic. Detailed checklists or procedures for the process of abstraction help identify the relevant system elements and properties. It is important that these checklists be based on a sound system understanding and generic principles of abstract system considerations, as discussed in chapter 3.1. The procedure described by STEINMEIER can be consulted as a first example: identifying system elements; system boundaries; internal relations; relations to the system environment; functions. and relevant properties [STEINMEIER 1999, p. 30]. However, the realization of these steps turns out to be the critical measure for success of system abstraction and the resulting assumptions. If system boundaries are defined

[68] The presented principles are also summarized well by FELGEN, yet insufficiently discussed and related to each other for the context of product architectur management (compare [FELGEN 2007, pp. 16 ff.]).

insufficiently, or relations or properties are neglected or overlooked, the assumptions based on the abstraction of the real system result in insufficient, or even misleading, system perceptions. To overcome these difficulties, the system abstraction is to be conducted gradually, i.e. top-down or bottom-up. The sufficient level of abstraction then depends on the goal of system analysis. The abstract description of a product's functions, an automotive drivetrain for example, can be derived on numerous levels, ranging from the overall description of the system function on the highest level of abstraction ("provide mobility") to solution neutral sub-functions ("store energy", "convert energy", "use energy") and, bottom-up, detailed functional descriptions ("transfer torque", "adjust rotation speed", etc.). Additionally, the functional abstraction can be conducted from different viewpoints, i.e. the abstraction of technical functions as shown, or from the perspective of the user ("drive conventional", "drive electric", etc.) [compare DEUBZER & LINDEMANN 2009b].

As a conclusion, the application of the fundamental principle of system analysis, abstraction, is accompanied by the following challenges: difficulties to identify relevant system elements, relations and properties, importance to identify the sufficient or adequate level of abstraction, and the need to take the right (stakeholder-) perspective of the system. All challenges are met by accompanying the process of abstraction during analysis with further principles, such as hierarchies for decomposition, element classes or systematic selectivity.

Hierarchy

The use of hierarchies when dealing with complex systems of any kind is frequently conducted to provide a system overview and support the human interactors with the system in their efforts to understand and describe the system sufficiently for different purposes [AHLEMEYER & KÖNIGSWIESER 1998, p. 22]. In the context of product architecture analysis, the goals of hierarchical decomposition are usually to gain system understanding to improve or renew products or parts thereof [CRAWLEY et al. 2004, KUSIAK 1999, p. 224]. In the context of manufacturing firms, the product decomposition, be it physical or functional, is used to structure the organization, manufacturing and development processes etc. [see e.g. DANILOVIC 2006, ENGEL & BROWNING 2008, GÖPFERT 1998]. Hierarchies are regularly modeled graphically in schematic manner, typically known from organizational charts. The aim and outcome of hierarchical decomposition is the structured breakdown of a system until reaching its "lowest level of elementary subsystem" [SIMON 1962]. The perception might differ as to which level is the lowest, depending on the viewpoint and purpose of the system modeling and analysis [SIMON 1962].

The effect of a hierarchical decomposition has to be discussed critically. Firstly, the depiction of hierarchies implies simplicity that does not actually represent the respective system. Complex systems comprise of numerous networked interrelations among elements that are not depicted in the hierarchical system representation.[69] Secondly, groupings of elements

[69] SIMON differentiates between the "formal organizational hierarchy", in which "authority relations" interrelate a subsystem and the system it belongs to, and hierarchies of complex systems, in which the relations are more complex and the relation of subordination does not exist or is at least less dominant [SIMON 1962, SIMON 1996, p. 185].

according to the given hierarchy are, at best, giving hints on the real clusters of the system structure. To give an example, the decomposition of product components can be conducted top-down according to the physical structure, starting with the largest unit and gradually decoupling the mechanical linkages down to smallest parts, where no further decoupling can be accomplished. Other strategies might focus on a decoupling, according to the fulfillment of functions, manufacturing processes or the fulfillment of requirements, assignment to organizational entities etc. While all of these strategies are valid for certain goals and situations, the early identification of the "right" way to decouple the system requires a detailed knowledge of the expected outcome, the approach and the inherent entities to consider. A change of perspective, from physical interrelations to functions for example, as is often perceived during the process of design, results in a situation where a hierarchy established before is useless. Decisions based on the simplified hierarchical view often cause problems later in the process, when the unconsidered interrelations and views point at the lacking comprehensiveness of the established model.

On the plus side, the simplified view of complex systems is sufficient in situations where only one view of the system is required to represent and quickly grasp the elements of a system. Further positive effects of hierarchical decomposition include the forced discussion of different existing levels of abstraction of a system. Whereas other methods, such as the functional models discussed in chapter 5.5.2, require a certain and defined level of abstraction, the hierarchical decomposition follows the actual degree of user knowledge. During the application of other models, practice has shown that users tend to mix different levels of abstraction due to their current state of knowledge about the system. The decomposition into hierarchies supports the process of system analysis and the complete perceptions of a system, given that the views of the system and existing levels of abstraction are known and defined.

Encapsulation

Encapsulation in software engineering is described as the dissociation of static from dynamic elements. As a goal of the application of the encapsulation principle, a subjective simplification of the object under consideration is strived for. This is to be achieved by hiding interior system events, which are not relevant for the analysis of the overall system ("information hiding" [Coad & Yourdon 1994, p. 30], compare also to the "Black Box" principle [Lindemann 2009]).

Even in this short description, similarities, particularly those concerning the difficulties in application, show with the principle of abstraction. In both principles, decisions must be made regarding which entities of the system have to be considered relevant or irrelevant. While the principle of abstraction considers the whole system on a unique level of abstraction, encapsulation, or the black box principle, allow for certain system elements or defined system areas to be considered on a different level of abstraction. Encapsulation can thus be considered to be a combination of the principles of abstraction and selectivity, while even hierarchies play a role if different system areas are modeled on different levels of abstraction.

As a result, encapsulation faces two of the three major difficulties of the principle of abstraction, i.e. the identification of relevant and irrelevant entities, and the adjustment of the sufficient or required level of abstraction. An additional challenge, and typical of the encapsulation or black-box principle, is the correct modeling of the behavior of the hidden system area; the information from this is not displayed in the overall model. More precisely, input and output information of the encapsulated system area have to reflect reality in order to allow for correct results of system analysis. Encapsulation, on the other hand, supports difficulties in strictly hierarchical decompositions, as not all elements have to be considered on the same level of abstraction. The model does not necessarily need to present all system entities on the same or all levels of abstraction. Again, the importance of different principles in combination shows through the example of encapsulation.

Scale

The principle of scale does not, as the colloquial meaning suggests, represent a mere variation of size, but according to COAD & YOURDON considers the trilateral relationship between the system, its parts, and the viewer or user of the system model [COAD & YOURDON 1994, p. 33]. The principle of scale is thus based on the identification and modeling of numerous distinct levels of abstraction. The system and its parts are differentiated according to the identified levels and set in relation to the viewer, i.e. different stakeholders requiring different levels of abstraction. A model assembled according to that notation and procedure enables the user to sufficiently work with and navigate through a complex system model [COAD & YOURDON 1994, p. 33].

The principle of scale, set in the context of the previously discussed principles, demands the recognition and identification of different levels of abstraction and their relation to different stakeholders. As such, the principle of scale combines the principles of abstraction and hierarchy to overcome challenges arising from the application of just one of these principles. Additionally, the principle of scale brings the viewer (or different stakeholders) into the process of analysis as an important component of the process of analysis.

Selectivity

Selectivity as a principle implies that the information of a complex system as a whole cannot be grasped as one, but rather is reduced according to the motivational problem of the system analysis. The resulting model is thus reduced to the relevant information, separated from the irrelevant [FUCHS 2004, p. 18]. Abstraction, as discussed earlier, focuses on taking the outside view of an object or reducing the information to one defined and consolidated level. Selectivity, on the other hand, separates certain views, rather than levels of abstraction; an example of this is analyzing a product by taking functions and form into consideration, but ignoring material, weight and other properties. Different semantics and notations of modeling languages and techniques imply this selectivity through the given rules and possible data to be stored in the model [compare FUCHS 2004, pp. 152ff.]. The constraints of the model's possibilities already imply selectivity, which is dependent on the type of model chosen and thus not influenceable.

The importance of selectivity as principle and its role as an enabler was pointed out during the discussion of the principles of abstraction and encapsulation. Selectivity alone is always accompanied by a high level of risk, implying that relevant factors are labeled as irrelevant and thus neglected in the considerations of analysis. It is thus recommended to recognize all parts of a system by the means of scale and encapsulation, thus not neglecting certain system elements completely but considering them with less detail [FELGEN 2007, p. 17]. The risk of missing the influences of irrelevant elements is thus reduced and allows for the purposeful analysis of the system.

AHLEMEYER & KÖNIGSWIESER state that selectivity is rather a necessity than an option, assuming that selectivity must be applied for the analysis of a complex system to be able to grasp the system at all. Again, the risk is pointed out, and thus the ambivalence of the principle, as "wrong" selection, causes insufficient and misleading results [AHLEMEYER & KÖNIGSWIESER 1998, p. 26-27].

In the context of other principles of analysis, selectivity proves to be rather an enabler of the combined application of other principles, than a principle to be applied solely. Yet, the demand for selectivity for complex system analysis is given at any point, as analysis implies the concentration of the relevant system elements for the reaching of reasonable conclusions.

Classes

The identification of classes means establishing a hierarchy of classes, combining similar elements of a system to classes. Similarity, that context describes elements with similar properties and corresponding behavior. The identification of classes as a principle demands the detailed description of classes and objects, their commonalities and differences. Even more, the definition of classes and objects should be enforced to reach a concise system perception [COAD & YOURDON 1994, p. 31].

The definition of classes enables a different view of systems than is possible with flow-oriented or hierarchical models By departing from these views, new possibilities for the combination and detachment of e.g. functions become apparent, enabling differentiated perspectives of the system and thus leading to new solutions [FELGEN 2007, pp. 17-18].

During analysis, the neglecting of the relations between elements and the establishment of hierarchies of classes facilitates the perception of the system. This effect is caused by the subjective reduction of the number and variety of system elements and relations. According to the definition in chapter 3.1, those have a major influence on complexity, thus making a more convenient system perception possible.

Of course, the presented application of the principle of classes can only be used as a first step in system analysis, as the neglecting of elements or objects and their interrelations is not an option for successful system analysis.

Association

The principle of association is briefly described as the combination of conceptions due to their similarity, and thus builds the foundation for the identification of classes and the establishment of hierarchies of classes. Association interrelates comparable objects of systems due to their similarity in behavior (time) or properties (physical) [COAD & YOURDON 1994, p. 31].

Despite its importance for analysis, association is also an important means for design synthesis and is applied as creativity method in product development. The abstraction of a problem is thus actively associated with similar problems in nature, technology or everyday life to enable the adaption of solutions from other areas into the problem solving process [compare LINDEMANN 2009]. The principle of association thus not only stands for a principle of system analysis, but also points to the close nature and interdependence of analysis and synthesis in design.

Conclusion

The discussion of different fundamental principles points to a number of conclusions discussed in this section. First, the origin of principles stems largely from areas of science such as sociology, economy, or software engineering, where the systems approach and systems thinking, as well as the human perception of systems, are more elaborately considered in the context of system analysis and handling complexity.

Second, the definition and description of the different principles show that similarities exist similarities between the principles, their goals and their difficulties in application. The discussion of the principles pointed to these challenges and similarities in detail. Nevertheless, the different principles allow for a varying approach to system analysis, and are thus all equally relevant, pointing to the third conclusion.

As the similarities and challenges of the different principles point out, a principle of analysis cannot stand for itself. For successful system analysis, different principles are required in combination, leading to a more complete and sufficient model of the system. The principles are applied sequentially, as well as in parallel in such an approach, ranging from the mere modeling of systems to a detailed behavioral or structural analysis. Examples were given in the detailed discussion of principles.

As a fourth conclusion, the advice for application of principles can be considered by stating that the awareness of the different existing principles allows for a more systematic analysis process by purposely varying the viewpoint of analysis through the application of different principles. The knowledge of which parts of the system were successfully decomposed, which encapsulated and on what level of abstraction, which presumptions during selectivity caused these results and from which stakeholder perspective etc. leads to a more profound system understanding. The results of system analysis can be discussed and judged more purposefully and trustfully than without the knowledge of existing analysis principles.

In addition to the discussed principles, based on the selection by FELGEN [FELGEN 2007, pp. 17ff.], further principles can be considered. These were excluded due to either their lack of generality, as they combine other principles, or due to their lack of relevance for the preceding work. Worth mentioning are the principles of communication (i.e. the handling of interfaces), behavioral categories (i.e. differentiation of behavior according to the differentiation of classes and objects) [COAD & YOURDON 1994, p. 32], object-oriented analysis (OOA) in general [BOOCH 1994, COAD & YOURDON 1994], the three step approach of Luhmann [AHLEMEYER & KÖNIGSWIESER 1998, p. 26-27], and several approaches in product development, which usually include numerous principles, yet combine them within a procedure as well as a modeling approach.

5.4.3 Networked system analysis

The discussion of different models, and possibilities of modeling in general, point out the relevance of matrix-based modeling techniques in the context of product architecture management. Matrix-based methods and modeling are very common occurrences in the different fields of product architecture management, i.e. identification of requirements, system analysis, synthesis and evaluation, as well as downstream activities.

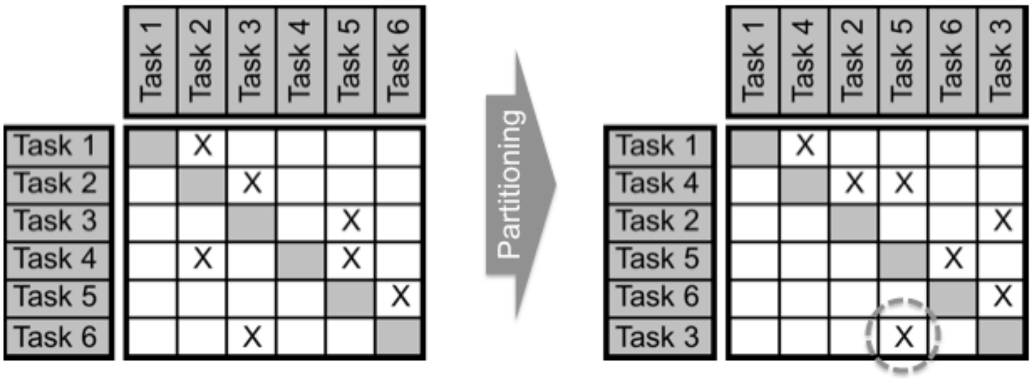
While these different existing approaches are discussed in chapter 5 in general, this section will cover the approaches used for system analysis, more precisely the analysis of complex system structures, defined as the product architecture in the beginning of this work. As the most elaborate and scientifically sound approach, the Design Structure Matrix was introduced, refined, and structured, both in terms of a procedure of application and as a classification of use cases over the past 30 years [compare STEWARD 1981, BROWNING 2001, MAURER 2007, pp. 53-64]. The following paragraphs will focus on the evolution of the DSM approach in the past decades, rather than give an overview of the different existing approaches using matrix-based techniques, as done by MAURER, but include the more recent multiple domain approaches, as introduced in his work [MAURER 2007, pp. 53-64]. While the analysis criteria and metrics are largely based on Graph Theory and Network Science, those areas of science are not considered more intensively, as discussed in chapter 3.2.3, due to the reasons outlined there, i.e. lack of general applicability for product architectures and scientific proceeding in general.

Matrix-based analysis approaches

Before the **DSM** became synonymous with elaborate measures in system analysis in general [BROWNING 2001], early use of the method was motivated by process analysis, redesign and optimization. As one of the first applications, STEWARD used the DSM to identify iterations in design processes. In the first application of the Design Structure System, the considerations are based on the logical interdependence of tasks. Tasks are then logically interdependent if parameters determined in one task are required in another [STEWARD 1981]. The typical representation of a system is conducted by matrix representation, in which the elements of a system are represented as rows and columns, and their interdependencies as marks in the matrix. The matrix is always a square intra-domain matrix, i.e. the diagonal contains no marks and the elements appear in the same order, both in columns as well as rows. STEWARD introduced the first methods, partitioning and tearing, whose application results in the DSM, an optimized structure of the design process in which the number of iterations is reduced to a minimum and remaining iterations are clearly pointed out and reduced to a minimum [STEWARD 1981].

The process of **partitioning** aims for the reordering of rows and columns in a fashion that allows for all marks of the DSM to be positioned on one side of the matrix, thus iterations do not occur. If a reordering of all marks on one side of the diagonal is not possible, which is usually the case, the remaining marks are positioned as close to the diagonal as possible, thus minimizing iterations [STEWARD 1981, MAURER 2007, p. 231]. Other names for the process of partitioning are sequencing or triangularization [MAURER 2007, p. 231].

Tearing follows the process of partitioning by identifying the marks that prohibit the partitioning process from fully completing, and removing these marks [Steward 1981, Maurer 2007, p. 140, Kreimeyer 2010, p. 55]. Of course, the removal of marks represents a change of the system, thus measures to implement this change in the real system have to be considered, though they often do not exist. To cope with remaining iterations in processes, for example, the assignment of senior staff is proposed by Steward to provide educated guesses based on experience when needed, i.e. parameters are required but still undefined for the next task [Steward 1981]. To enable the identification of most promising tear marks, the application of other algorithms, such as the identification of feedback loops in the system can be conducted [Maurer 2007, p. 217], as partitioning algorithms might not lead to the best results when overlapping feedback loops occur [Maurer 2007, pp. 105-106].



	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Task 1		X				
Task 2			X			
Task 3					X	
Task 4		X			X	
Task 5						X
Task 6			X			

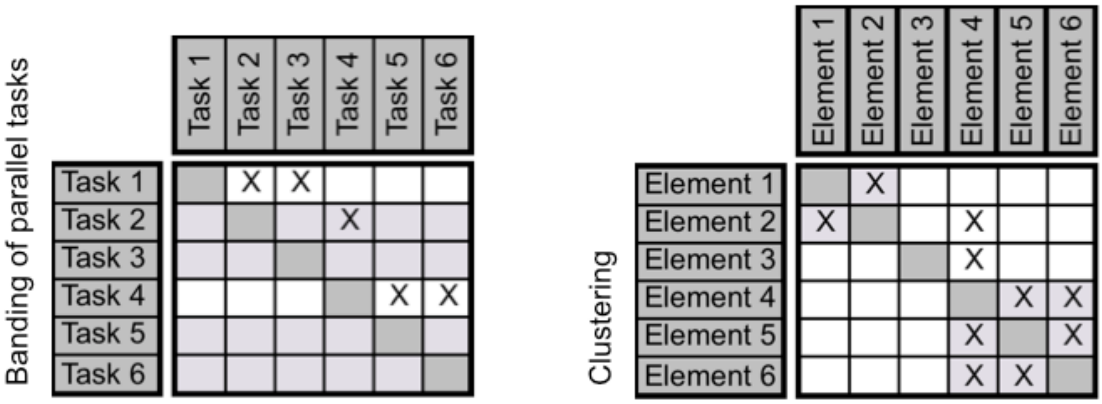
	Task 1	Task 4	Task 2	Task 5	Task 6	Task 3
Task 1		X				
Task 4			X	X		
Task 2						X
Task 5					X	
Task 6						X
Task 3				X		

Figure 5-6 Process of partitioning and possible mark for tearing (encircled)

After the first use in process management, DSM application spread through the different areas of design science, resulting in what is classified by Browning as component- and people-based DSM, as well as parameter- or task-based DSM [Browning 2001], depending on what is depicted by the model. Application is thus possible for engineering design, process optimization, organizational issues or basically any system describable by parameters. In accordance with the different DSM-classes, further methods for matrix-optimization evolved. Worth mention are the methods of clustering and banding, which are usable with different types of systems.

Banding represents the rearrangement and markup of elements in the matrix, so that "parallel entities remain" [Kreimeyer 2010, p. 55]. As a result, the marked elements, for example tasks, can be executed in parallel, while components, on the other hand, might be designed in parallel. Parallel entities in the domain of components do not directly influence one another, although of course indirect interdependencies may occur.

Clustering, as a later evolved means of matrix analysis, helps identify elements that are mutually related and thus densely connected with one another, yet loosely connect with other system elements [MAURER 2007, p. 227, KREIMEYER 2010, p. 55].[70] The clustering method is frequently applied to identify modules in products. The common definition of modules is underlying, stating that modules are represented by highly interconnected elements with few interrelations outside of the module [ULRICH 1995].


Banding of parallel tasks
Clustering

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Task 1	▨	X	X			
Task 2		▨		X		
Task 3			▨			
Task 4				▨	X	X
Task 5					▨	
Task 6						▨

	Element 1	Element 2	Element 3	Element 4	Element 5	Element 6
Element 1	▨	X				
Element 2	X	▨		X		
Element 3			▨	X		
Element 4				▨	X	X
Element 5				X	▨	X
Element 6				X	X	▨

Figure 5-7 Banding (parallel tasks highlighted) and clustering (clusters highlighted)

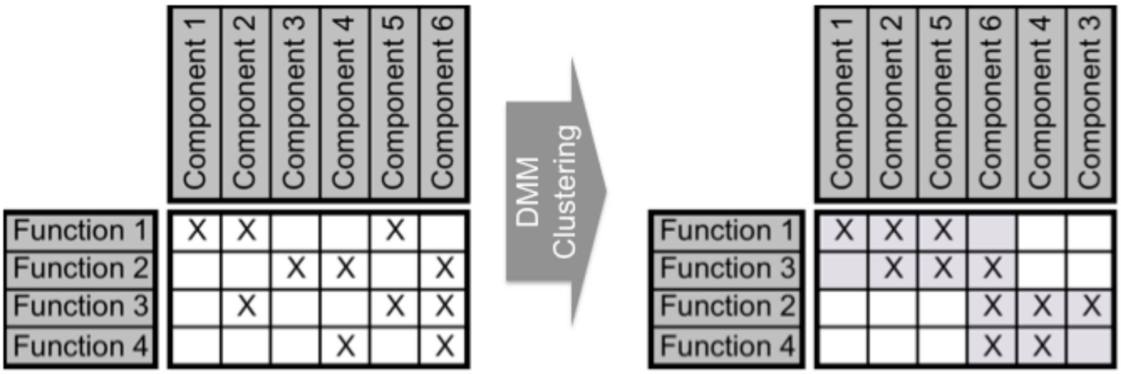
The main premise of the DSM approach and the above-discussed methods is the representation and model of the system. A uniform model is required for the discussed types of analysis, i.e. the methods demand elements and interrelations of the same type for meaningful, i.e. interpretable, analysis results. As the discussion in the introductory chapter 1.1 pointed out, complex systems are characterized by a large number and an even larger variation of different and numerous elements and interrelations. As a result, the DSM can only depict a small cutout of the actual system, which was identified as relevant for system analysis. Though PIMMLER & EPPINGER point out different dependency types for products (i.e. spatial, energy, information, material) which are incorporated into a comprehensive product model [PIMMLER & EPPINGER 1994], other authors include e.g. user and environmental interfaces [YASSINE & WISSMANN 2007]. The interpretation of structural characteristics, e.g. clusters in the resulting architecture, are difficult to interpret, whether for example spatial interferences have priority over information flows and so on. LINDEMANN et al. provide a similar example using the dependency type's spatial, functional and features, yet prioritizing the different dependency types to point out and differentiate between functional or physical modules etc. [LINDEMANN et al. 2009, p. 185].

Although many analysis tasks can be accomplished through concise modeling, there was growing interest in depicting the system and its interrelations across different domains. To cope with this issue, DANILOVIC & BROWNING put forward and established the concept of the **Domain mapping Matrix (DMM)**, i.e. a inter-domain dependency matrix contrasting two

[70] If clusters are largely overlapping, matrix representation is insufficient as the elements cannot be rearranged so that all clusters are identifiable in the matrix. The supplementary depiction in graph-form is therefore recommended [MAURER 2007, p. 104, compare SHARMAN & YASSINE 2007]

different domains with each other, e.g. product functionality and product components [DANILOVIC & SANDKULL 2004, DANILOVIC & BROWNING 2007].

To benefit from this system representation, the concept of clustering a DSM was transferred to the non-square dependency matrix, resulting in a **DMM-clustering**. The underlying concept of DMM-clustering is the acknowledgement of the similarity of rows and/or columns. As a result of application, rows and columns with similar marks are grouped and rearranged accordingly, pointing out potential clusters of elements.



	Component 1	Component 2	Component 3	Component 4	Component 5	Component 6
Function 1	X	X			X	
Function 2			X	X		X
Function 3		X			X	X
Function 4				X		X

	Component 1	Component 2	Component 5	Component 6	Component 4	Component 3
Function 1	X	X	X			
Function 3		X	X	X		
Function 2				X	X	X
Function 4				X	X	

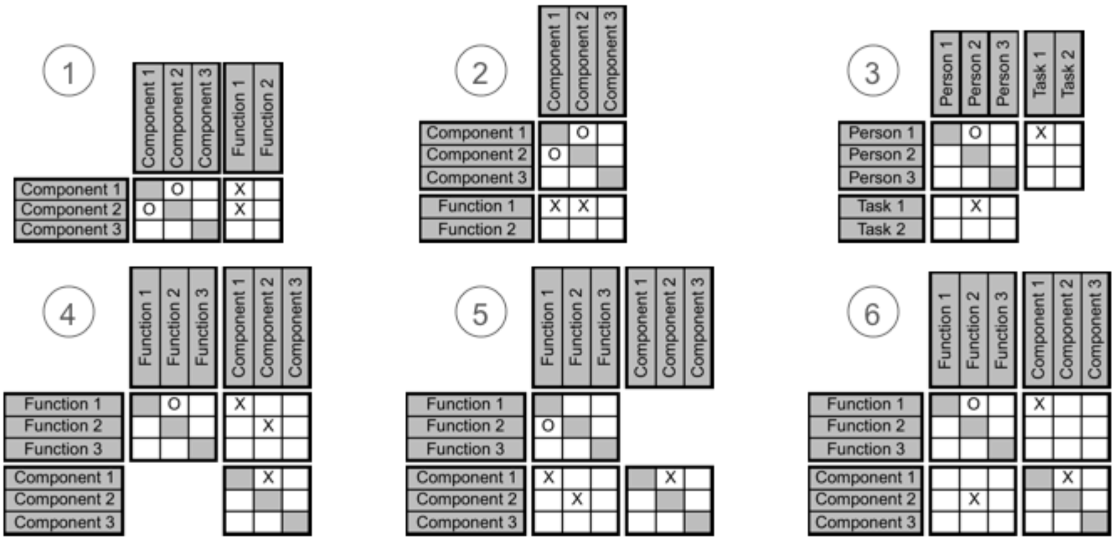
Figure 5-8 Process of DMM-clustering (potential clusters highlighted)

The introduction of the DMM as a complementary supplement to the DSM enables the depiction of systems from different angles, as well as the interaction of those systems. DANILOVIC & BROWNING give an example of the application by coupling five project domains (goals, product, process, organization, tools) and five product domains (requirement, functionality, parameters, specification, product), of which the product poses the link between the two [DANILOVIC & BROWNING 2007].

What DANILOVIC & BROWNING described as the “periodic table of DSMs and DMMs” [DANILOVIC & BROWNING 2007] was the multi-domain representation of a system, representing involved domains as well as the interrelations within and between the respective domains. Yet, the application of structural characteristics is limited to the clustering of elements within the DMM, while the analysis of a DSM through discussed techniques and the structural characteristics, which will be pointed out in the following sections, reveal a far larger potential than the mere clustering. To fully tap this potential, the **Multiple-Domain Matrix (MDM)** was introduced by MAURER, systematically decomposing a system into its domains and the interrelations between them, accompanied by techniques to cope with the resulting structures and making them accessible for thorough analyses [MAURER 2007, pp. 72ff.].

To achieve this accessibility, MAURER introduced **domain mapping logics** [MAURER 2007, pp. 112-116], mathematically described by the multiplication of matrices. By applying these logics, interdependencies across domains can be mapped onto the element of one domain, resulting in a DSM for thorough analysis. In fact, STEWARD, in his first DSM application, used a similar principle inexplicitly, stating that tasks are interrelated if the product parameters defined within the respective tasks are dependent on one another [STEWARD

1981]. Using the language of the MDM, the DMM depicting which task defines which parameter and the DSM of parameters influencing one another are transferred into the DSM of tasks. The six existing domain mapping logics and their mathematical and matrix representations are depicted in the following figure, in which each case also represents a multiple domain matrix.



Case	Source DMMs	Source DSMs	Equation
1	1	-	$DSM_{comp} = DMM_{comp/funct} \times DMM_{comp/funct}^{T}$
2	1	-	$DSM_{comp} = DMM_{funct/comp}^{T} \times DMM_{funct/comp}$
3	2	-	$DSM_{pers} = DMM_{pers/task} \times DMM_{task/pers}$
4	1	1	$DSM_{funct} = DMM_{funct/comp} \times DSM_{comp} \times DMM_{funct/comp}^{T}$
5	1	1	$DSM_{funct} = DMM_{comp/funct}^{T} \times DSM_{comp} \times DMM_{comp/funct}$
6	2	1	$DSM_{funct} = DMM_{funct/comp} \times DSM_{comp} \times DMM_{comp/funct}$

Figure 5-9 Possible cases of domain mapping logics to compute an aggregated DSM [MAURER 2007, pp. 112-116, see also KREIMEYER 2010, pp. 53-54]

To achieve this accessibility, MAURER introduced **domain mapping logics** [MAURER 2007, pp. 112-116], mathematically described by the multiplication of matrices. By applying these logics, interdependencies across domains can be mapped onto the element of one domain, resulting in a DSM for thorough analysis. In fact, STEWARD, in his first DSM application, used a similar principle inexplicitly, stating that tasks are interrelated if the product parameters defined within the respective tasks are dependent on one another [STEWARD

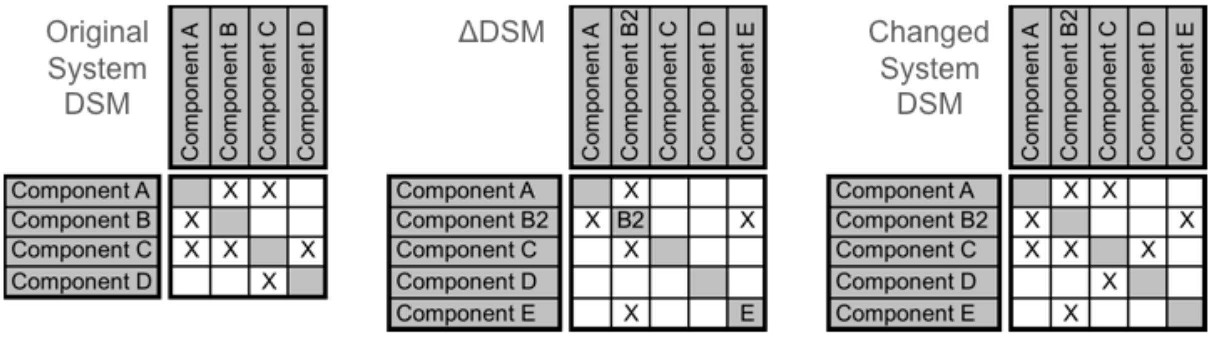
1981]. Using the language of the MDM, the DMM depicting which task defines which parameter and the DSM of parameters influencing one another are transferred into the DSM of tasks. The six existing domain mapping logics and their mathematical and matrix representations are depicted in the following figure, in which each case also represents a multiple domain matrix.

Techniques, characteristics and metrics

Based on the possibilities of the application of DSM and MDM, a number of methodologies, analysis criteria and metrics evolved. Typical analysis processes were discussed in chapter 5.4.1, of which that of PIMMLER & EPPINGER in particular deals with the application of DSM, while that of MAURER considers the general approach of complex systems in engineering design, based on the comprehensive use of MDM [PIMMLER & EPPINGER 1994, MAURER 2007].

To cope with the structural system models established within the analysis processes, several techniques, structural characteristics and metrics exist. The techniques of **partitioning, banding, tearing, clustering,** and **domain mapping logics** were discussed in the preceding sections. Additionally, few procedures exist, supporting the operation with the system models for different purposes.

For the area of design synthesis, which is naturally underrepresented among DSM-applications with a focus on system analysis, a number of techniques exist to support the process of synthesis. DE WECK et al. show an example of technology infusion into an existing system, introducing the **ΔDSM** to identify changes occurring in an existing system due to the involvement of new components. The ΔDSM is introduced to assess the invasiveness of new technologies in terms of a cost/performance trade-off during conceptual design [DE WECK 2007]. Of course, the generic application, as well as the abstract example in the figure below, can only point out the general potential. For a comprehensive and analytically correct approach, the marks in matrices are represented by numerical values, which then allow for positive as well as negative values, to both create and remove elements and relations. Analytically more correct, comprehensive, and systematical, the problem was approached by EBEN et al., showing that the ΔDSM concept allows for only four out of eight possible cases of change, i.e. the creation and removal of elements and relations, but lacking possibilities to deal with the merging and splitting of elements, replacing relations with one or more other relations, or replacing relations with an element [EBEN et al. 2008]. For the merging and splitting of elements, a DMM for mapping the changes was introduced by EBEN et al., but a comprehensive solution is still to be defined to solve all eight cases in a satisfactory manner [EBEN et al. 2008].



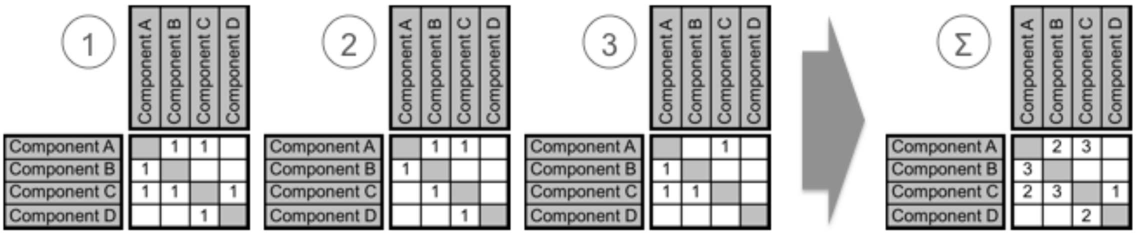
Original System DSM	Component A	Component B	Component C	Component D
Component A		X	X	
Component B	X			
Component C	X	X		X
Component D			X	

ΔDSM	Component A	Component B2	Component C	Component D	Component E
Component A		X			
Component B2	X	B2			X
Component C		X			
Component D					
Component E		X			E

Changed System DSM	Component A	Component B2	Component C	Component D	Component E
Component A		X	X		
Component B2	X				X
Component C	X	X		X	
Component D			X		
Component E		X			

Figure 5-10 Example of application of the ΔDSM: "Original System DSM + ΔDSM = Changed System DSM" [from DE WECK 2007]

From the contexts of variant management and design synthesis emerged a similar concept, motivated by insufficient visualization of different possibilities or paths within a solution space. BRAUN & DEUBZER proposed the addition of different matrices, each depicting one possible product variant as a complete cluster, to achieve as a sum the overall solution space in which every possible variant is represented by a completely interlinked cluster [BRAUN & DEUBZER 2007]. GORBEA et al. put forth a similar concept based on the analysis of different conceptual alternatives, which, as a result of addition, form the **ΣDSM**. More comprehensively, if different domains are considered, the model can be described as the ΣMDM. Thereby, many concept matrices are numerically added, resulting in a summation of possible alternatives in a comprehensive solution space. The numerical values of marks resulting from addition are considered, pointing at relations that tend to appear in all or only a few conceptual alternatives [GORBEA et al. 2008]. The difficulty is to identify necessary and sufficient relations to achieve a fully functioning system. A combination of both approaches was therefore utilized by HELLENBRAND & LINDEMANN for systematic synthesis, depicting a compatibility matrix based on existing alternatives and identifying further possible alternatives by the identification of completely interlinked clusters [HELLENBRAND & LINDEMANN 2008].



	Component A	Component B	Component C	Component D
Component A		1	1	
Component B	1			
Component C	1	1		1
Component D			1	

	Component A	Component B	Component C	Component D
Component A		1	1	
Component B	1			
Component C		1		
Component D			1	

	Component A	Component B	Component C	Component D
Component A			1	
Component B	1			
Component C	1	1		
Component D				

	Component A	Component B	Component C	Component D
Component A		2	3	
Component B	3			
Component C	2	3		1
Component D			2	

Figure 5-11 Example of application of the ΣDSM: Addition of three system matrices

The existence of structural characteristics and metrics to characterize complex systems is widely acknowledged, yet a sound classification and comprehensive overview for the context of engineering design has long been missing. MAURER gives an overview on a number of structural characteristics, summed up in the following table. MAURER therein differentiates between criteria for single nodes and edges, as well as subsets of the system. The listing additionally contains analysis criteria for graphs, which include, among others, the previously introduced techniques of banding, clustering and partitioning [MAURER 2007, pp. 197ff.]. Methods for system analysis are included, which utilize existing analysis criteria and set them into an operable context. For example, the structural ABC-analysis orders edges, in accordance with their appearance in feedback loops (or hierarchies, clusters etc.) for a better comprehension of critical system elements [MAURER 2007, p. 236]. Methods to optimize structures are considered separately, as they inevitably include a change to the system, in contrast to analysis methods, which solely discuss the system as is [MAURER 2007, pp. 238ff.].

Table 5-1: Structural characteristics and methods for analysis (as compiled by Maurer 2007, pp. 197ff.)

Analysis criteria for nodes and edges
Active and passive sum
Activity
Articulation node
Attainability
Bridge edge
Bus
Closeness
Criticality
Distance (global)
End and start node
Isolated node
Leaf
Transit node
Analysis criteria for subsets
Bi-connected component
Cluster (completely cross-linked)
Cluster (strongly connected parts)
Distance between nodes
Feedback loop
Hierarchy
Locality
Path
Quantity of indirect dependencies
Similarity
Spanning Tree
Strongly connected part/component

Analysis criteria for graphs
Banding
Clustering
Degree of connectivity
Distance matrix
Matrix of indirect dependencies
Partitioning (Triangularization, Sequencing)
Methods for system analysis
Feed-forward analysis
Impact check list
Mine Seeking
Structural ABC-analysis
Trace-back analysis
Methods for optimizing structures
Tearing
Evolutionary algorithms

Kreimeyer provides a comprehensive overview of metrics for the analysis of processes, based primarily on the findings of Graph Theory and Network Science, which in fact is partially overlapping with the identified characteristics and methods compiled by Maurer [Kreimeyer 2010, pp. 300ff., Maurer 2007, pp. 197ff.]. As the focus of the metrics lies on the analysis of processes, an adaptation to complex systems in general or product architectures in particular tends to be difficult. Processes are characterized by a large quantity of similar treatable tasks, persons etc., while the numerous different elements of product architecture, as well as their interactions, cannot in their sum provide a statistically equal basis for analysis.

Table 5-2: Metrics for structural analysis (as compiled by Kreimeyer 2010, pp. 300ff.)

Size and density
Number of domains
Number of nodes
Number of edges
Number of classes

Hierarchies
Height of hierarchy
Width of hierarchy
Tree criticality
Snowball factor

Number of interfaces between domains
Number of edges per node
Relational density
Number of unconnected nodes
Adjacency
Activity / Passivity
Degree correlation (nodes)
Degree correlation (edges)
Degree distribution
Fan criticality
Synchronization points / distribution points
Number of independent sets
Attainability
Number of reachable nodes
Reachability of a node
Closeness
Proximity
Relative centrality (based on between-ness)
Connectivity
Node connectivity
Edge connectivity
Paths
Number of paths
Path length
Weight of an edge
Centrality of path (based on centrality)
Centrality of path (based on degree)
Degree of progressive oscillation
Forerun factor
Tree-robustness
Maximum nesting depth
Clustering
Number of cliques
Cluster-coefficient (local)
Cluster-coefficient (global)
Module quality 1 (flow of information)
Module quality 2 (compactness)
Cycles
Number of cycles
Number of cycles per node
Number of cycles per edge
Number of feedbacks
Activation of cycle
Number of starting points for iterations
Iterative oscillation
Several domains
Bipartite density
Number of organizational interfaces
Cognition
Cognitive weight
Degree of non-planarity
Boolean Operators
McCabe Cyclomatic Number
Control-Flow Complexity
Log-based Complexity

Based on the close relation Graph Theory, the discussed techniques, characteristics and metrics are often represented in matrix-notation for mathematical processing, yet accompanied by graph representations for visualization and communication. The coupling of the DSM-approach and Graph Theory is largely supported by Maurer and other authors [Collins et al. 2008, Kreimeyer 2010, Maurer 2007].

To conclude, techniques (or methods), characteristics and metrics can be utilized for the analysis of complex systems. Techniques can be characterized by a change of visualization and a rearrangement of the system elements. The characteristics point to certain patterns within the architecture, of which the relevance and meaning must be interpreted in each use case. Finally, metrics result in numerical values for nodes, edges or groups, which are especially useful in statistically relevant systems like processes.

The challenge for product architecture management is to identify which of the above-mentioned patterns are suitable for each respective domain, and how results are to be interpreted. The compilations above, besides contributing greatly to the science of complex systems in engineering, allow for a systematic analysis by providing checklists for the analysis process and leading to an improved perception of the complex system in question.

Fields of application and capabilities

Based on the previous discussion, the following fields of application for the MDM approach can be identified, as has been proposed by BROWNING: the process, the product, the organization, and parameterized models [BROWNING 2001].

However, the application underlies the following constraints, divided into shortcomings of the notation and shortcomings in practical application.

Shortcomings of the notation

- DSM application is restricted to the view of one system type (i.e. “domain”, according to MAURER 2007) with defined type of interrelations.[71]
- Though MDM allows for the consideration of different domains, a defined level of abstraction is still required for application within intra- and inter-domain matrices. Systematic guidance is missing regarding how to cope with different levels of abstraction.
- The appropriate handling of hierarchical decomposition accompanies the coping with abstraction and is yet unsolved [DANILOVIC & BÖRJESSON 2001, KREIMEYER 2010, p. 58].
- Changes over time are considered by different authors, as discussed by KREIMEYER, yet not sufficiently solved [KREIMEYER 2010, p. 58 compare DE WECK 2007, EBEN et al. 2008]. The handling of decision points in networks as a further step was recently researched [KREIMEYER 2010, pp. 123ff.].
- The management of both conceptual variants and variants within the product portfolio was considered using matrix-based approaches, yet a comprehensive approach is missing, especially one employing a combination of the two [DEUBZER & LINDEMANN 2009a, DEUBZER & LINDEMANN 2009b, GORBEA et al. 2008, HELLENBRAND & LINDEMANN 2008].

Shortcomings in practical application

- Practice has shown difficulties in application for untrained users of DSM, DMM and MDM approaches. Though users in engineering design are accustomed to matrix and accompanying graph representations [MAURER 2007, p. 109], tapping the full potential is particularly difficult for untrained users.
- Information acquisition processes are highly demanding and crucial for the outcome of any method. The problem is well-known and different approaches are introduced, yet systematically largely unsupported [MAURER 2007, pp. 94ff.].
- The introduced generic methodologies, techniques and applications are to be adapted for each use case. In few cases is it possible to apply the core ideas without extensions, adaptations and trade-offs.

[71] The combination of different interrelation types has to at least be considered critically [compare the example of LINDEMANN et al. 2009, p. 185]

- As it is intended for analysis, bridging the gap to design synthesis from DSM and MDM is still a challenge. Though measures are known to induce creativity for solution finding in a systematic way [MAURER 2007, pp. 135ff.], a coupling with systematic methods of synthesis is still missing, hindering a combined use of existing methodologies [DEUBZER & LINDEMANN 2009c].

Some of the issues mentioned above are also discussed from the perspective of process analysis by KREIMEYER, where solutions are provided for the area of process analysis and optimization, which are not easily transferable to product architecture management [KREIMEYER 2010, pp. 57-58].

Conclusion

The methods and approaches, which evolved from the DSM to the DMM and MDM, have proven to be powerful in numerous projects. In fact, there is no generic analysis technique for the analysis of complex structures that is structured and capable yet easily applicable in engineering design. Especially in comparison to matrix-based synthesis methods, such as axiomatic design [SUH 2001, pp. 10ff.] or conceptual design synthesis [KUSIAK 1999, pp. 243ff], the efforts in analysis seem to be far more generic in their possible applications, as well as more elaborate in their means.

Research is on the way to addressing a number of shortcomings recently: enabling the combination of analysis and synthesis [WYATT et al. 2008, DEUBZER & LINDEMANN 2008, GORBEA et al. 2008, HELLENBRAND & LINDEMANN 2008], refining analysis methods (e.g. in cooperation with control engineering [DIEPOLD et al. 2009]), developing guidelines for practical application and interpretation of characteristics [e.g. KREIMEYER 2010 for processes] or treating different fields, such as variant management [BRAUN & DEUBZER 2007, DEUBZER et al. 2008] or scenario analysis with existing methods.

The presented work will utilize existing approaches and enhance them, enabling them to deal with product architectures in different phases. Focus is then placed on the interlocking of analysis and synthesis, as well as the interpretation of known characteristics in the context of product architectures.

5.5 Product architecture definition and synthesis

The definition and synthesis of product architectures has become one of the major challenges in engineering design. The numerous disciplines involved in engineering developed large volumes of knowledge, which differ increasingly in the use of language, standardized procedures and processes, as well as undergoing paradigm shifts due to innovation, regulations etc. Due to this “burst of knowledge” [EHRLENSPIEL 2009, p. 19], the knowledge about the complete system becomes more important, yet more difficult to grasp for the individuals involved. Research in systems engineering based on the understandings of systems science in particular (compare chapter 3.2) aims at solving these shortcomings. Nevertheless, it is the cooperation of disciplines in research, as well as industrial practice, that puts forth the most promising and innovative solutions [EHRLENSPIEL 2009, p. 19].

Given these recent developments, the situation of individual designers developing complete systems from scratch is no longer the norm. In the automotive industry, for example, 60 to 80 % of parts are newly designed, while the rest are used from predecessors or other models of the current portfolio [CLARK & FUJIMOTO 1991, p. 148]. HUBKA & EDER claim that even 95 to 99 % of all design problems are concerned with redesigning [HUBKA & EDER 1996]. As a resulting premise, the design of products has always incorporated the fact that, from the beginning of development onwards, the maturity or concretization and level of detail of the respective product entities differ largely from one another. The discussion of existing methods in the following section will show that they support the designer in an individual situation of design on a certain level of maturity, but cannot easily be carried over to all tasks of synthesis of complex products or easily be integrated into the process of systems architecting.

The following abilities have to be considered when adapting a system, according to CHMARRA et al.:

- Ability to recognize the change from an old environment to a new environment
- Ability to determine the change that has to be made to the system (according to the recognized change in the environment)
- Ability to effect the change to generate the new system [CHMARRA et al. 2008]

Against that background, ROOZENBURG & EEEKELS differentiate between four views of synthesis [ROOZENBURG & EEKELS 1995, pp. 4-9]:

- Synthesis (and analysis) as a **phase of the design process**, i.e. a subsystem of the (now outdated) view of a linear process of product concretization [compare VDI 2221]
- Synthesis as **part of the problem solving process**, in which “analysis” can be understood, both as the clarification of the task based on the actual situation before synthesis, as well as the deduction of consequences of a synthesized scenario [compare LINDEMANN 2009, PONN & LINDEMANN 2008]
- Synthesis as **assembly of subsystems**, i.e. the application of e.g. the morphological chart to problems which can be decomposed based on the system’s inherent flows [compare EHRLENSPIEL 2009]
- Synthesis as **integration of ideas**, based on different sub-problems and views, such as different Design for X aspects e.g. assembly, manufacturing etc. [compare e.g. BOARDMAN & SAUSER 2008]

Within the design process, the problem solving process can be applied for detail design or conceptual design, as well as any other occurring “problem” in design or entrepreneurial activity. The assembly of subsystems, as well as the integration of ideas, can both be considered as tasks of systems engineering (or systems architecting). The synthesis of single design tasks and systems engineering are inseparable from one another both temporally and logically. As a result, reactions from both areas have to be considered with their respective counterpart, for which the method for the management of product architectures has to provide appropriate means. The discussion of the state of the art in synthesis reflects this challenge and discusses correspondingly critical methods.

To structure the state of the art in engineering design synthesis, methods are differentiated, according to EHRLENSPIEL, into conventional methods, creativity-supporting techniques (intuitive procedures), and systematic approaches (discursive procedures), which include methods for combining solutions [EHRLENSPIEL 2009, pp. 400ff., PAHL et al. 2007, pp. 77ff.]. To complete the picture for product architecture management, functional modeling and automated approaches are added. Functional modeling is commonly described as a starting point for design synthesis, which EHRLENSPIEL describes as "structuring the design task" [EHRLENSPIEL 2009, pp. 390-400, compare PAHL et al. 2007, pp. 145ff.]. Functional models aim for bridging the gap between a vaguely described requirements situation and the technical solution to clarify the task. Automated procedures ("computational synthesis") were developed recently and are based on rules or grammars to synthesize product architectures with unfortunately little generic achievements up to now.

5.5.1 Process of product architecture synthesis

The process of synthesis differs according to the chosen method and approach. Computational synthesis varies significantly from manual procedures, due to the less time-consuming solution generation of automated procedures, where solutions can be generated according to rules very quickly. The challenge, therefore, is the evaluation of the different solutions' properties as a basis for decision-making. Manual solution finding procedures, on the other hand, require a guided process of continuous concretization with decision steps, to enable less time-consuming processes. As a result, the manual process of solution finding results in a limited number of feasible and elaborate solutions, while the outcome of automated procedures shows a large number of solutions, greatly differing in feasibility, but aiming for the depiction of an almost exhaustive solution space.

Furthermore, it is of importance to differentiate why problem solutions or design alternatives are sought. The morphological chart [ZWICKY 1969], for example, aims at the purposeful and comprehensive combination of solution principles to achieve functional and proper solutions for the complete system [EHRLENSPIEL 2009, pp. 428-431]. Brainstorming, on the other hand, is intended and suitable for problems that are precisely outlined, yet not too complex; this is for finding not only technical solutions, but also for any sort of problem during the development process [EHRLENSPIEL 2009, p. 406]. Accordingly, the need for a precise description of how to apply the method differs, as does the required time and effort to conduct method application. In the following sections, methods for synthesis are analyzed, whether they are suitable for the definition of architectures or only for the identification of solutions for single functional problems.

An overview on methods of synthesis can be found in classical engineering design and business management books [PAHL et al. 2007, p. 127, PONN & LINDEMANN 2008, EHRLENSPIEL 2009, ULRICH & EPPINGER 2003, pp. 100 ff.], as well as specialized synthesis overviews from different authors [as provided by ANTONSSON & CAGAN 2001, CHAKRABARTI 2002, CAGAN et al. 2005].

The process of synthesis is described by ULRICH & EPPINGER in the following steps [ULRICH & EPPINGER 2003, pp. 100ff.]:

- Clarify problem (understand, decompose, focus on critical sub-problems)
- Search externally (lead users, experts, patents, literature, benchmarking)
- Search internally (individual, group)
- Explore systematically (classification tree, concept combination table)
- Reflect on solution and process (feedback)

MAHER identifies three steps for synthesis [MAHER 1990]:

- Decomposition
- Case-based reasoning
- Transformation

The process of synthesis, in the context of automated design synthesis, is described by CAGAN et al. as [CAGAN et al. 2005]:

- Representation
- Generation
- Evaluation
- Guidance

TERPENNY & MATHEW focus on the required models for design synthesis and order the process accordingly [TERPENNY & MATHEW 2004]:

- Functional model
- Solution model
- Component model

PAHL et al. identify six steps, necessary for design synthesis [PAHL et al. 2007, p. 127]:

- Confrontation
- Information
- Definition
- Creation
- Evaluation
- Decision

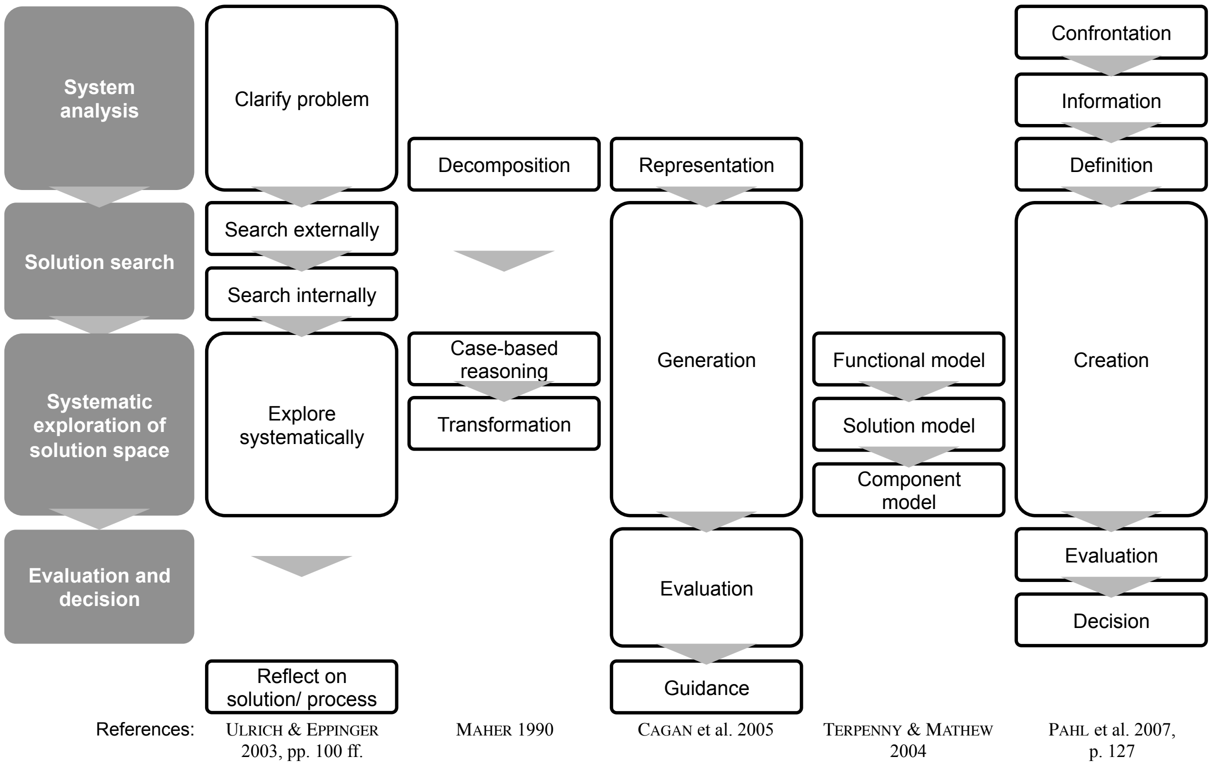


Figure 5-12 Process of synthesis (left) derived from existing approaches

TOMIYAMA & SCHOTBORGH point out a number of distinct circumstances, which have to be taken into account regarding the special requirements of the synthesis of product architectures. . Product architecting is a **multidisciplinary activity** performed by a team composed of experts from various domains. The product architecture defines the **boundaries of mono-disciplinary activities**, which in the following have to be **coupled by systems integration technology**. The integration of disciplines represents **not only the summing up the elements of design**; in contrast to the mere grouping of components, the **overall functionality** of the product architecture is defined [TOMIYAMA & SCHOTBORGH 2007]. The following sections discuss methods for design synthesis and compare those to the requirements and circumstances of product architecture synthesis.

5.5.2 Functional modeling as a prerequisite

Functional modeling per se does not pose a method for design synthesis or the definition of product architectures. Nevertheless, many authors recommend functional modeling as the basis for solution finding in the early stages of design processes, especially in classical approaches of engineering design research [e.g. EHRLENSPIEL 2009, pp. 390-400, LINDEMANN 2009, pp. 117ff., PONN & LINDEMANN 2008, pp. 53ff., PAHL et al. 2007, pp. 169ff., ULRICH & EPPINGER 2003, pp. 101ff., STONE & WOOD 2000]. The successful application of functional modeling was also verified by experiments [KURFMAN et al. 2001]. Later developments of computational synthesis aim for a similar application of functional models, especially if the definition of product architectures is the focus. CAGAN et al., for example, define function as well as form as necessary prerequisites for computational design synthesis, represented in the form of graphs and rules or grammars [CAGAN et al. 2005, KURTOGLU & CAMPBELL 2006, WOOD & GREER 2001, p. 220]. Different functional models evolved over the years, inheriting semantics and product representations before the realizations of concepts or physical representations of the products exist. Stemming from the understanding of design as a process from function to form [CAGAN et al. 2005], most aim to support of creativity, for example the relational functional models known of the TIPS[72] methodology or elementary hierarchical models [LINDEMANN 2009, pp. 117ff.].

STONE & WOOD sum up the achievements through functional modeling for design as follows [STONE & WOOD 2000]:

- First of all, the functional modeling is inevitable for **product architecture development**. Especially in the early phases, when important decisions are required for product modularization and the assessment of possibilities across a wide product portfolio, functional considerations are formative for the product architecture.
- The **systematic function structure generation** is supported by an agreed upon set of functions and flows, eliminating the need for different models for different disciplines. The model as such provides a unique representation and supports the later definition of physical models.
- Based on this unique representation, the functional representation, if kept generic, can serve as an **archive and transmitter of design information**. Not only can product information be communicated more easily based on a common model, but the information can also be stored sustainably throughout processes and different development projects.
- For comparing products, both within the company's portfolio as well as for the benchmarking with competitors, functional models act as a valid basis for the **comparison of product functionality**. The fulfillment of newly arising requirements and needs can be mapped onto existing products of similar functionality. This is one of the applications of a functional model acting as information archive.
- Functional modeling supports **creativity in concept generation** by aiding the decomposition of the design task. Based on the formal abstract description, important decisions can be made in the early phases of design.
- Being a high-level physical representation of the product, functional models help formulating objective measures in terms of **product metrics, robustness, or benchmarks**. These measures can be used for benchmarking and quality endeavors.

A number of existing functional models will be discussed in the following sections, giving an overview of the different views provided by renowned authors of the field [compare FUCHS 2004, pp. 24-26 and ERDEN et al. 2008 for overviews]. Additionally, the models are

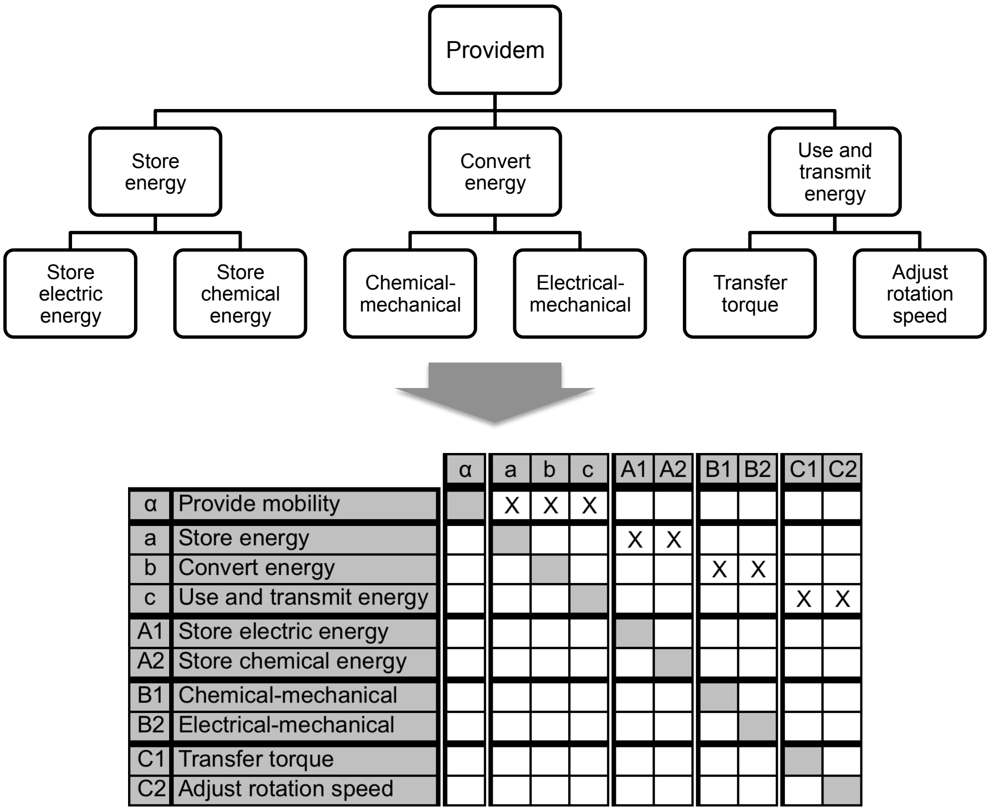
[72] Theory of Inventive Problem Solving (TIPS), also referred to according to the Russian acronym TRIZ [CAVALLUCCI et al. 2002]

transferred into a generic modeling method based on the discussion in chapters 4.2 and 4.3. As such, the link between the requirements model, the functional model and the models in the following sections can be provided. The functional models considered in this overview represent product-functions only, i.e. modeling languages depicting the functions of processes, such as the Structured Analysis and Design Technique (SADT), or manufacturing functions, as in the Integration Definition for Function Modeling (IDEF0) [FUCHS 2004, p. 195, KUSIAK 1999, pp. 2-7] are not discussed. Functional modeling is also a prerequisite for different automated procedures [WOOD & GREER 2001, p. 220], which will be discussed in chapter 5.5.7 about computational synthesis.

Hierarchical functional model

According to LINDEMANN, a hierarchical structuring of product functions is generally possible, to limited extent even for flow-oriented, relational or user centered functional considerations [LINDEMANN 2009, p. 119, compare PAHL et al. 2007, pp. 170ff.]. Functional decomposition in hierarchical form is intended to provide insights for analysis and understanding based on the system approach [ULRICH & EPPINGER 2003, pp. 101-103]. The up- and downsides of hierarchical structuring during analysis, as discussed in chapter 5.4.2, are of course valid for hierarchical functional decomposition.

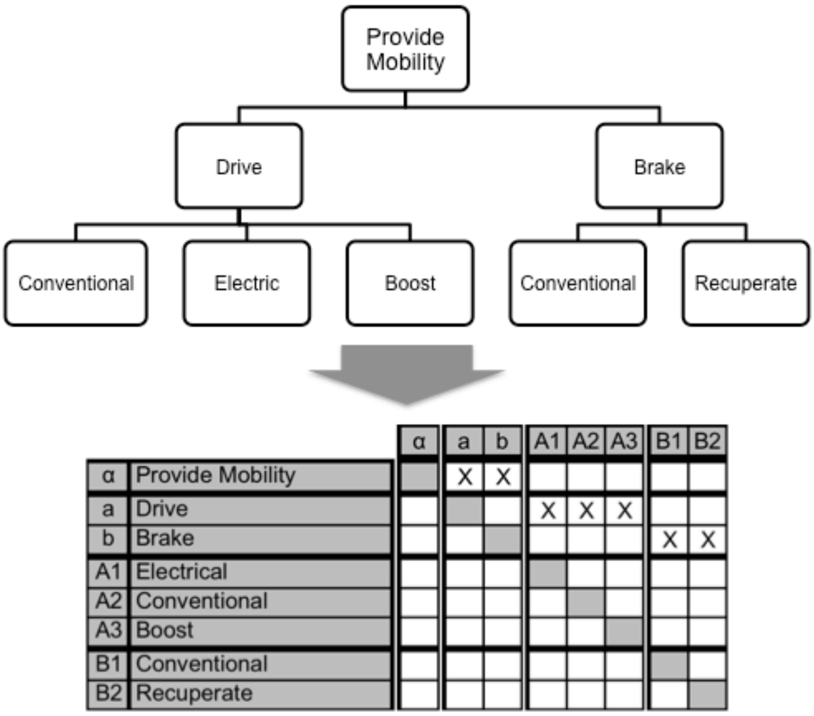
DEUBZER & LINDEMANN provide an example of hierarchical functional decomposition, in which the positive aspects, as well as negative aspects, of hierarchical structuring are shown. In the given example, the functions of an automotive drivetrain are decomposed. Therefore, two different viewpoints were chosen, that of technical (or system) functions and that of user functions, i.e. functions actively used, chosen and experienced by the user [DEUBZER & LINDEMANN 2009b]. The example pictorially shows the ambiguity of hierarchical decomposition, as the different viewpoints can hardly be integrated or coupled in hierarchical form, yet both show important views of the product architecture. The following figures show both hierarchies more elaborately adapted from [DEUBZER & LINDEMANN 2009b]. To cope with functional hierarchies in the chosen modeling approach, the identified hierarchies are modeled in matrix-form as an example, using the MDM approach. Thereby, the distinct levels of the hierarchy, as well as the branches, are modeled separately as a respective domain.



		α	a	b	c	A1	A2	B1	B2	C1	C2
α	Provide mobility	▨	X	X	X						
a	Store energy		▨			X	X				
b	Convert energy			▨				X	X		
c	Use and transmit energy				▨					X	X
A1	Store electric energy					▨					
A2	Store chemical energy						▨				
B1	Chemical-mechanical							▨			
B2	Electrical-mechanical								▨		
C1	Transfer torque									▨	
C2	Adjust rotation speed										▨

Figure 5-13 Hierarchical functional decomposition of an automotive drivetrain (technical view)

In the above figure of a functional decomposition from the technical perspective, the functions are classified according to their purpose. Other possibilities for hierarchical function decomposition include an "is required for" decomposition or "is realized by" decoupling of functions. EHRLENSPIEL describes these circumstances with the example of the evolution of a functional model during the development process, where certain decisions, for example the decision for a combustion engine and against an electric motor, cause the necessity of follow-up functions, e.g. the provision of fuel, which can then be displayed hierarchically [EHRLENSPIEL 2009, p. 396].



		α	a	b	A1	A2	A3	B1	B2
α	Provide Mobility	▨	X	X					
a	Drive		▨		X	X	X		
b	Brake			▨				X	X
A1	Electrical				▨				
A2	Conventional					▨			
A3	Boost						▨		
B1	Conventional							▨	
B2	Recuperate								▨

Figure 5-14 Hierarchical functional decomposition of an automotive drivetrain (user view)

The positive effect of hierarchical decoupling is the quick overview to grasp a system, given that a defined “paradigm” or classification for the hierarchy is existent and agreed upon. Challenges remain nevertheless, especially if different paradigms coexist. Functional units or modules are represented by branches, if the modeling paradigm is of the “is required for” type [PONN & LINDEMANN 2008, p. 62]. The presented modeling in matrix representation is intended to enable the interrelation of coexisting paradigms within the overall method.

The downsides can be easily retraced in the given example, as discussed in the section on hierarchies as analysis principle in chapter 5.4.2. The following listing sums up the shortcomings of hierarchical functional decomposition:

- The hierarchy can only depict **one perspective** of the system; the numerous complex interdependencies have to be neglected, yet are important, as the following functional models will show.
- Visualized groups, or branches of the hierarchy, depend largely on the chosen paradigm of the hierarchy, and as such cannot represent the **clusters** existing in reality, as are revealed through DSM-application, for example.
- To choose the “right” **paradigm** for hierarchical decomposition depends largely on the situation and goals of analysis. As such, the choice represents the major challenge of hierarchical analysis.
- The **change of perspective** or change of paradigm cannot be easily conducted, and inevitably results in parallel models, which are difficult to interrelate or are not interrelated at all.

Flow-oriented functional model

The flow-oriented functional model proposed by EHRLENSPIEL [EHRLENSPIEL 2009, pp. 390-400] is well known in engineering design research, and allows for the analysis of systems by consideration of information-, material- and energy-flows from a functional perspective [as proposed in PAHL et al. 2007, p. 32]. Being formulated as neutral as possible to technical solutions [LINDEMANN 2009, p. 120], the functional model provides support to design synthesis based on the differentiation of a product's functions [EHRLENSPIEL 2009, p. 401].

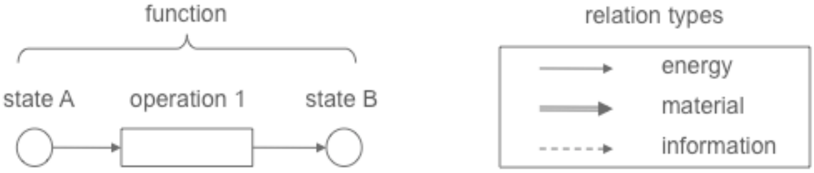


Figure 5-15 Semantics of the flow-oriented functional model [according to EHRLENSPIEL 2009, p. 398, PONN & LINDEMANN 2008, pp. 64-66]

The above figure illustrates the semantics of the flow-oriented functional model. A **function** describes the purposeful change of the input-state of a material-, signal- or energy-flow to an output-state. A **state** represents the sum of the flow's current properties at the respective point. The **operation** describes the actual process responsible for the change of state. States and operations are coupled with **relations**, which are divided into material-, energy- and information-flows [EHRLENSPIEL 2009, pp. 390-400].

The major benefit of the presented model is the functional decomposition of a complex product, based on the flows of material, energy and information. As a downside, products that do not focus on flows of material, energy or information within their functional structure, or do not contain functions of that sort at all, are hard to fully grasp using the given methodology. In practice, it is other flows that are depicted. Those include the flow of force and supporting forces between static parts for example, or the flow of air in pneumatic systems. However the full potential of the method cannot be tapped in that way. Further shortcomings, according to PONN & LINDEMANN, are the insufficient depiction of dynamic changes over time and the limited number of available logical operators e.g. to visualize decisions [PONN & LINDEMANN 2008, pp. 63-64].

In the following figure, a principle transformation of the flow-oriented functional model from schematic visualization to matrix representation is conducted, as proposed by DEUBZER & LINDEMANN, by establishing domains for each respective entity of the model (states and operations in the given example) [DEUBZER & LINDEMANN 2008]. In step one, the interdependencies between operations and states are deducted and represented in the respective DMMs of the notation. Second, the relations between states are calculated through domain mapping logics (case 3, compare with chapter 5.4.3). The same principle allows for the calculation of the relations between operations in the third step [DEUBZER & LINDEMANN 2008].

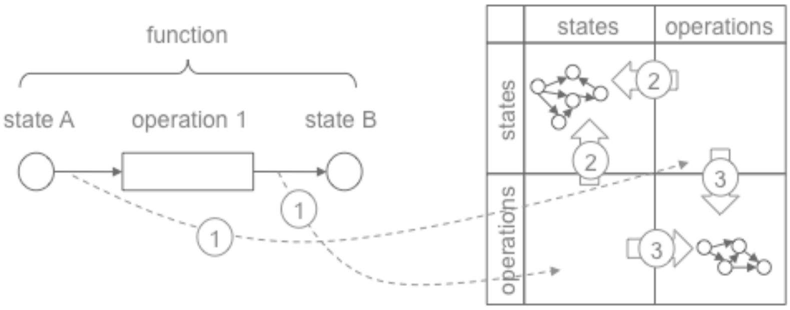


Figure 5-16 Transformation of the flow-oriented functional model into matrix representation [DEUBZER & LINDEMANN 2008]

To complete the transformation of flow-oriented functional models, different relationship types (energy, material, information) can be depicted in separate DMMs for a more thorough analysis. An example of this kind of application was already cited from LINDEMANN et al., where different relationship types were viewed distinctly and prioritized for modularization purposes [LINDEMANN et al. 2009, p. 185].

Relational functional model

The relational functional model, or relation-oriented functional model, offers a distinct additional view, especially when compared with the functional models discussed in the previous sections. While other models consider functions as positive, useful, or desired system behavior (or the translation of requirements into product behavior), the relational functional model introduces harmful or undesirable functions as an accompanying viewpoint of the product [compare LINDEMANN 2009, p. 120], which are one of the causes of behavioral complexity, as defined in chapter 3.1.3. The relational functional model is thus most appropriate for the analysis of existing systems and their inherent functional elements and relations [LINDEMANN 2009, p. 119], but, as such, can only contribute to the improvement of existing systems, rather than support new product development, i.e. the development and design of products from scratch.

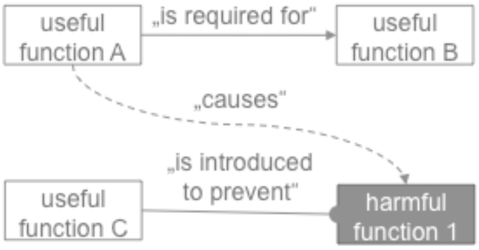
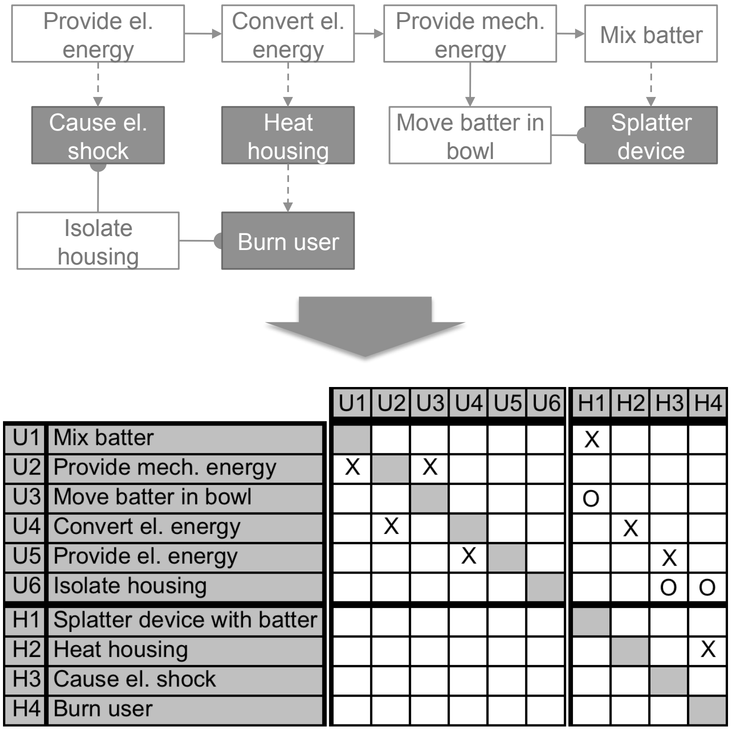


Figure 5-17 Semantics of the relational functional model [compare with LINDEMANN 2009, p. 120]

In the given example, depicted to explain the semantics of the model, function B represents the desired function of the system, for which function A is required; however, this causes the undesired function 1. To compensate for these shortcomings, function C was introduced to prevent the negative effects of function 1 from arising.

The hierarchical functional model, as was discussed, requires a paradigm to be conducted purposefully and concisely, while the flow-oriented functional model is based on the correct depiction of the flows within the system. The establishment of the relational functional model, on the other hand, follows a number of steps to achieve a thorough system representation. The procedure proposes the main function (or one of the main functions) as a starting point, followed by the respective closest useful and harmful functions.

The result of this procedure is the depiction of an existing system, showing its useful behavior as well as inherent flaws or harmful effects and behavior. The resulting structure can then be analyzed to identify conflicts to be solved (e.g. provide certain useful functions without causing harmful effects) and support creativity by introducing suggestions based on checklists to solve inherent conflicts [for details see PONN & LINDEMANN 2008, pp. 331 ff.]. An example of a relational functional model is provided in the following figure, representing a hand mixer system, and additionally introducing the transformation in matrix notation. The relations of the DSM of useful functions represent “is required for” relations, while relations of the type “causes” are marked “X” in the useful-harmful DMM and the DSM of harmful functions. Relations of the type “introduced to prevent” are marked “O” in the matrix.



		U1	U2	U3	U4	U5	U6	H1	H2	H3	H4
U1	Mix batter							X			
U2	Provide mech. energy	X		X							
U3	Move batter in bowl							O			
U4	Convert el. energy		X						X		
U5	Provide el. energy				X					X	
U6	Isolate housing									O	O
H1	Splatter device with batter										
H2	Heat housing										X
H3	Cause el. shock										
H4	Burn user										

Figure 5-18 Example of the relational functional model [LINDEMANN 2009, p. 120] and matrix notation

The most valuable benefit of the relational functional model is its view of negative functions, providing a more thorough analysis of the system, in comparison to other functional models. The application of the model enables the identification of flaws within the existing system, pointing out potential for improvement and prioritization during development. As practical application in industry has shown, the establishment of relational functional models supports the interview process of knowledge carriers in a structured manner, leading to a more complete picture than other methods with less structuring could provide. The interviewer is enabled to ask the right questions and supported in structuring the results by the semantics at hand.

As a downside, the relational functional model cannot be used as a substitute for other functional models to link to the technical solution, which is more elaborate in the flow-oriented functional model, for example, and is not easily to be provided. Therefore, it is missing the relation to the system components, which are implicit in the harmful functions of the relational functional model.

User-centered functional model

The user-centered functional model provides yet another viewpoint from a functional perspective of the system. While the other functional models focus on the system itself and the inherent functionality – useful or harmful – and behavior, the user-centered model depicts the application of the product under different circumstances, according to the numerous existing users along the product lifecycle [LINDEMANN 2009, pp. 118-119].

While the model cannot actively or systematically support the process of identifying relevant stakeholders along the lifecycle, the product use by the complying stakeholder during the respective lifecycle phase is depicted. The Unified Modeling Language provides the means to model a use case, which in fact is only a small portion of the UML methodology [TERPENNY & MATHEW 2004].

5.5.3 Conventional methods

The existence of conventional methods for design synthesis can be described above all from the perspective of the engineering design of less complex products. For those, solutions can be found “conventionally”, in a sense, as the main function or functions can be solved and realized through already existing solutions, known as abduction [TOMIYAMA & SCHOTBORGH 2007] or TOMIYAMA et al. 2003]. EHRLENSPIEL accordingly describes conventional methods as approaches where solutions are found from sources which provide existing solutions, which in the following sections can be overtaken or adapted, based on the experience of the designer [EHRLENSPIEL 2009, p. 404]. The term “solution”, instead of product, is used intentionally at this point, since it is usually certain modules such as valves, motors or gears that are chosen [LINDEMANN 2009, p. 138], rarely fully functioning and complex products.

While it is logical to first identify the obvious existing solutions to certain design problems, it is just as obvious, considering the sources of conventional solutions, that the challenge of systems architecting cannot be solved through the use of the conventional approach as the combination of solutions, i.e. the coupling to an overall product architecture, is still unsolved by these means.

The above-stated shortcomings of conventional solution finding become even more obvious when considering the possible sources of solutions, both company internal and external [EHRLENSPIEL 2009, p. 404, ULRICH & EPPINGER 2003, pp. 104ff., ZANKER 1999, p. 130]:

- Existing solutions within the company (product portfolio, bills of material, experts)
- Existing solutions from suppliers (catalogues)
- Existing flaws and solutions from customers (lead users)
- Literature research (specialist journals, specialist books, research reports, databases)
- Solutions in patents (patent reviews)
- Solutions from competitors (catalogues, trade fairs, product benchmarks)

If, for example, a flow-oriented functional model as discussed in chapter 5.5.2 exists, solutions for each function can be identified from sources cited above to contribute to a solution space, which in theory provides a solution to the overall problem by combining the individual solutions. However, the architecting of the product, i.e. the definition of the coupling of elements, can only be achieved manually by detailing the combination of solutions. A more comprehensive solution space from the architectural perspective, based on the given solutions, can only be achieved by systematic variation of the functional model, i.e. functional integration, separation etc. (for a comprehensive overview see e.g. [PONN & LINDEMANN 2008, p. 300], compare also the principles of modularization as summed up by [CHMARRA et al. 2008]).

As a conclusion based on the previous discussion, the conventional methods for solution finding support the establishment of a solution space in terms of separate solutions for individual problems, yet cannot sufficiently support the process of systems architecting [TOMIYAMA & SCHOTBORGH 2007]. The following sections will discuss to what extent other methods and approaches can contribute to the definition and variation of architectures.

5.5.4 Creativity-supporting techniques

Since the designer cannot rely on existing solutions available through conventional methods in all cases of design, creativity is required for the definition of novel solutions during problem solving cycles. To be creative during design, i.e. producing something new, or relating known principles to novel problems, is, though highly relevant, a mostly intuitive process. Creativity-supporting techniques aim for the provision of an environment that supports this creative process [EHRLENSPIEL 2009, pp. 404ff.]. While EHRLENSPIEL states that between 50 and 100 different techniques exist [EHRLENSPIEL 2009, p. 406] and the absolute number is likely to be higher, the following sections will discuss only the characteristic properties of creativity-supporting techniques in general, of which e.g. association or synectics are popular representatives [ULRICH & EPPINGER 2003, pp. 109f.]. As a result, the adequacy of creativity-supporting techniques for product architecture synthesis is evaluated.

Certain requirements are necessary to be fulfilled in order to access the design problem using creativity-supporting techniques. First, the clarification of the problem is a necessary prerequisite for the application of creativity-supporting techniques, i.e. define and question requirements, decompose the problem into smaller sub-problems, and focusing on the most critical aspects of a problem, or the most promising aspects from a customer perspective. It is absolutely necessary to meet the core of the problem to achieve solutions that generally suit the problem. Goals and requirements should be quantified, if possible [ULRICH & EPPINGER 2003, pp. 100-104].

Equally important is the visualization of solutions (developed e.g. in brainstorming sessions or the gallery method). Other participants can generate ideas based on the solutions proposed by others. The organized transfer of solutions to other participants for further idea generation is systematized by other methods as well, such as the method 6-3-5 [EHRLENSPIEL 2009, pp. 406-407].

In general, the number of solutions is initially more relevant than the quality, the evaluation and selection. The decision-making process follows after the creativity sessions. The same is valid for the grounding of solutions, which appear too distant from the concrete problem. Nevertheless, solutions off the beaten track are desirable [EHRLENSPIEL 2009, pp. 404ff].

Criticism causes restraints for further solution generation among practitioners, and is thus undesired in group sessions of any kind. Association and analogy is not only welcome, but even supported by certain methods (e.g. method 6-3-5) or focused on in certain fields, such as nature (e.g. in bionics). Also supported is the use of unrelated stimuli to encourage new ideas based on photographs or objects more or less related to the task [EHRLENSPIEL 2009, pp. 404ff., ULRICH & EPPINGER 2003, pp. 109-110]. From a psychological point of view, the change of perspective, supported e.g. by certain procedures and models, is important, since new solutions are mostly generated on the basis of existing solutions from other fields of applications [KNOBLICH 1997, p. 214-215]. Even systematic analogies are proposed by some authors, for example through automated means [e.g. QIAN & GERO 1996, applying Function-Behavior-Structure (FBS) for system comparison and analogy].

The systematized problem clarification through functional modeling was discussed in chapter 5.5.2. Special mention should be made in the context of creativity-supporting techniques to the relational functional model, as the basis for the TIPS methodology. The formulation of the functional model, including harmful functions, provides possibilities for systematically expressing conflicts in existing solutions [PONN & LINDEMANN 2008, pp. 86-88]. To solve these expressed oppositions, the TIPS methodology provides checklists containing principles to systematically access the problems [PONN & LINDEMANN 2008, pp. 331ff.]. The overall methodology of TIPS contains further means; compare e.g. the works of CAVALLUCCI et al. [CAVALLUCCI et al. 2002].

In the context of product architectures, creativity-supporting techniques as discussed promise only limited success. Precise problem formulations and the problem decomposition are required for the application of the methods; complex architecture issues as discussed cannot comprehensively be tackled. Creativity-supporting techniques are preferably used for (detail) design problems, rather than composing and managing product architectures. Nevertheless, the application can contribute to definable problems in the overall context of product architectures, yet cannot substantially support the framework. The following sections discuss methods more suitable and systematic methods for an overall approach.

5.5.5 Systematic approaches to creative problem solving

Based on the discussion of creativity-supporting techniques, the shortcomings of unsystematic approaches became apparent. Systematic approaches strive for a more rational and methodical application of design knowledge. The foundation of this application is the existence of rules to establish a system and existing building blocks to be combined by the rules [TOMIYAMA & SCHOTBORGH 2007]. These building blocks can consist of one of the following:

- Components (collected as discussed in the section "Conventional methods")
- Working principles (compare e.g. bond-graphs as in [THOMA 1975])
- Functions and systematic functional variation (compare chapter 5.5.2 and e.g. [PONN & LINDEMANN 2008, p. 300])
- Attributes (functional requirements and design parameters [SUH 1990])

The foundation of systematic approaches is the modeling of the above-mentioned building blocks and the provided stimuli of the designer's creativity. Existing solutions can be analyzed and improved, based on these representations [TOMIYAMA & SCHOTBORGH 2007].

The advantages of systematic approaches, which intend to compensate for the shortcomings of conventional and creativity-supporting techniques, are not solely based on the representation. ULRICH & EPPINGER sum up further reasons for systematic approaches. The listing includes downsides that can potentially occur when synthesizing unsystematically [ULRICH & EPPINGER 2003, p. 99]:

- Conventional methods advance the consideration of only few obvious solutions, but fail to systematically support a more **comprehensive and persistent identification of the solution space**.
- Careful **analysis of solutions from competitors and other industry branches** seldom takes place within the application of conventional or creativity-supporting techniques. Systematic procedures support this behavior.
- Systematic procedures foster the **involvement of interdisciplinary teams**.
- Integration of promising partial solutions into a **fully functional architecture** is not supported by the conventional methods for solution finding and thus requires systematic support.
- Since systematic procedures offer a **framework for solution finding**, entire categories of solutions can be systematically explored, while otherwise individual categories could potentially not be considered due to less obvious suitability.

For a comprehensive overview of systematic approaches in design synthesis, EHRLENSPIEL comprehensively identifies five approaches: structuring schemes; design catalogues; checklists of physical effects; structuring schemes for technical contradictions; and checklists in general [EHRLENSPIEL 2009, pp. 409ff.].

Structuring schemes aim for the structuring of partial solutions of the solution space generated through the application of conventional methods or creativity-supporting techniques. Structuring schemes include one-dimensional structuring schemes, such as the morphological chart [Ehrlenspiel 2009, p. 410, Ritchey 1998, Zwicky 1969] or the classification tree [Ulrich & Eppinger 2003, p. 112].

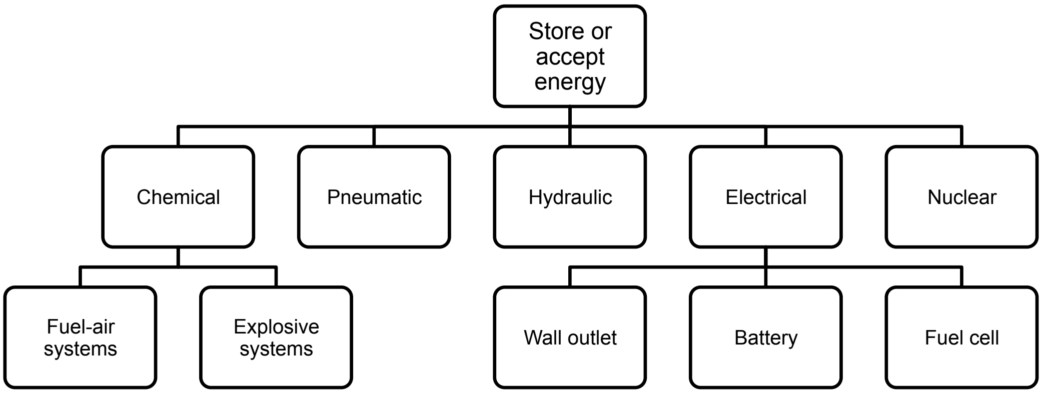
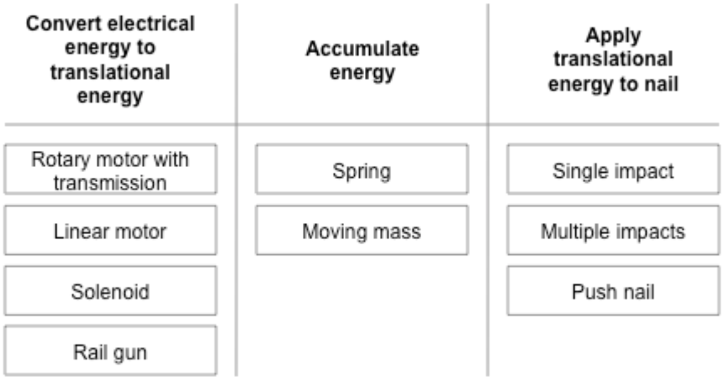


Figure 5-19 Example of the classification tree [Ulrich & Eppinger 2003, p. 112]

While the classification tree represents a structuring of general solutions, such as the physical effects for a single function "Store or accept energy", as depicted above, the morphological chart represents a visualization of solutions for partial functions of the system and can also be represented as a concept combination table [Ulrich & Eppinger 2003, pp. 114ff.].



Convert electrical energy to translational energy	Accumulate energy	Apply translational energy to nail
Rotary motor with transmission	Spring	Single impact
Linear motor	Moving mass	Multiple impacts
Solenoid		Push nail
Rail gun		

Figure 5-20 Example of a concept combination table for a nail gun [Ulrich & Eppinger 2003, p. 114]

The approach of the concept combination table, as well as the morphological chart, intends to combine partial solutions to comprehensive product concepts based on the structured depiction of the solutions (for a procedure to cope with structuring schemes see [Ehrlenspiel 2009, pp. 428-432]). Due to the systematic composition, the application of morphological charts is accessible to automated approaches, resulting in a support of solution finding [Ritchey 2006, Ölvander et al. 2009].

Design catalogues [e.g. Roth 2001] represent structured collections of solutions, which are in many cases structured as multi-dimensional structuring schemes. The goals of design catalogues are summed up by Ehrlenspiel as:

- Quick, function-oriented access to solutions

- Provision of a possibly complete solution space
- Identification of suitable solutions based on attributes and properties
- Support of automated procedures

Solutions are accessible through the functional characteristics of solutions and their properties, attributes and tolerances, which are systematically structured in the design catalogue [EHRLENSPIEL 2009, pp. 415-416].

Checklists of **physical effects** are structured similar to design catalogues, though they represent more generic solutions. As the main purpose, physical effects intend to support the creativity of designers and the finding of innovative solutions (comprehensive overviews on the composition and application of checklists for physical effects can be found in [EHRLENSPIEL 2009, pp. 417-426, PONN & LINDEMANN 2008, pp. 81-85]). Comprehensive and structured physical effects are presented by PONN & LINDEMANN [PONN & LINDEMANN 2008, pp. 311ff.].

Based on the relational functional model (chapter 5.5.2), **structuring schemes for technical contradictions** offer solutions for problems, which appear to be unsolvable at first glance. The procedure includes the steps of formulation of the technical problem, the problem abstraction, which results in a functional description based on standardized formulations, a search for stimuli based on the pairings of functions and finally a solution finding based on stimuli from the matrix of contradictions [PONN & LINDEMANN 2008, p. 87 and 331ff. compare TOMIYAMA & SCHOTBORGH 2007].

Checklists are intended to stimulate creativity, similar to physical effect lists, and can be applied during the complete problem solving cycle, depending on the information included in the checklists. If comprehensive checklists are available, the risk of neglecting solutions and possibilities can be minimized [EHRLENSPIEL 2009, p. 427]. Checklists are best established to be product- and company-specific and formulated to be objective, complete and comprehensible [EHRLENSPIEL 2009, p. 428]. A number of comprehensive checklists for general application during design are provided by PONN & LINDEMANN, some of which have been already referred to in the previous sections [PONN & LINDEMANN 2008, pp. 291ff.].

As a concluding remark, systematic approaches for design synthesis provide procedural guidelines as well as models, and improve design synthesis in terms of the reuse of technical solutions and stimuli for creativity. The advantages are those of a systematic procedure in general, i.e. to be quicker, more comprehensive and thorough, and enabling a focus on the most relevant aspects of the respective task.

Further approaches, such as axiomatic design or the consistency matrix, were only included as side notes and are discussed more exhaustively in the following section. For the purpose of product architecture management, the discussed approaches can be differentiated into methods to support creativity and methods for the systematic structuring of the solution space. Both types can overlap for certain methods, as is the case for the morphological chart or design catalogues.

For the management of product architectures, the discussion of the two types of approaches results in acknowledging that the questioning and enrichment of the solution space of individual solutions is as important as the composition of the architecture. Both tasks require systematic support and are strongly interwoven. Therefore, they need to benefit from the knowledge and results of their respective counterpart.

For design synthesis, components, working principles, functions, and attributes emerged as the main domains to cope with and achieve innovative solutions, given that requirements are clarified beforehand. To identify which method of application in which domain is most likely to support incremental or breakthrough innovations appears to be unanswerable for generic use. The synthesis on component- or physical-effect level can cause changes to the architecture, resulting in breakthrough innovations, as well as the synthesis on the architectural level, based on known principles and components. This underlines the importance of keeping track of the interrelation of the whole and its parts, i.e. the architecture as well as the subordinate elements.

5.5.6 Matrix-based synthesis methods

Methods using matrices are very common for system analysis, the Design Structure Matrix (DSM) above all (compare chapter 5.4.3). Related methods applying matrices similarly, such as QFD, have proven to be powerful measures when analyzing existing systems as well (compare e.g. chapter 5.6.3). For design synthesis, on the other hand, the mentioned methods and approaches were only able to make a small contribution, especially when dealing with new, not yet existing, solutions, rather than redesign. They appear to be more of a foundation or starting point for synthesis, rather than measures to support the process or outcome of synthesis.

The latest approaches, such as the Multiple-Domain Matrix (MDM), allow for the integration of different domains. That in mind, the MDM can depict functional models and their relations, as well as their connectivity to different physical principles, desired states of a product, attributes, components, requirements etc. Based on the modeling of the aforementioned entities, the assumption is made that the repeatedly iterative process of analysis and synthesis can be supported by the existing matrix-based modeling approaches.

Given that a systematic approach is required, and the interrelations between the single entities, as well as the overall architecture, are highly relevant for synthesis, the understanding of the linkage between the whole and its parts can be fostered through the application of matrix-based modeling and analysis approaches.

Researchers are aware of the gap between design synthesis and the analysis with matrix-based modeling approaches. Several authors reduce the existing limitations by expanding the method. The following sections discuss approaches to designs synthesis, based on the DSM- and MDM approach and other matrix-based approaches in general. Some of those expansions were already discussed in chapter 5.4.3, such as the ΔDSM and ΣDSM.

De Weck et al. introduce the **ΔDSM** to model and analyze the difference between the actual state and the novel design state on a parts-level. This approach depends on a product as a starting point, specified changes to the product and a physical representation of the novel product, at least based on change propagation during the conceptual state [De Weck 2007].

GORBEA et al. add the functional perspective in the spirit of the MDM approach and add up numerous existing solutions in a **ΣDSM,** similar to the variant management approach introduced by BRAUN & DEUBZER [GORBEA et al. 2008, BRAUN & DEUBZER 2007]. Given both the functional and component perspectives, a widening of the solution space is achieved within the range of existing solutions through adding up the existing products.

Both approaches allow for the analysis of a theoretical solution space based on existing solutions. At the same time, the introduction of novel solutions or the identification of improvement by the introduction of new technologies is not well supported, due to the close relation to existing solutions.

WYATT et al. introduce an approach of **automated synthesis based on DSM and DMM** application. Typically for automated synthesis, a large number of solutions are generated, in the given example airplane architectures, and existing DSM analysis methods are applied to evaluate and compare the generated architectures. As a major novelty, WYATT et al. also use matrices for the storage of information regarding how the architecture can be composed in advance. The connection requirements are represented in a DMM, depicting the ports of elements, i.e. what can be connected and how many times [compare KUSIAK 1999, p. 224]. The path requirements reflect the functions, which have to be fulfilled by the architecture [WYATT et al. 2008]. The path "well to wheel" in automotive drivetrain development is a typical example of a path requirement representing a function, as is described by DEUBZER & LINDEMANN [DEUBZER & LINDEMANN 2009c]. Though based on existing components, the presented approach allows, within these limitations, for the design of novel solutions.

In the same spirit, the **consistency matrix** depicts possible combinations of existing elements. Though used in scenario management [LINDEMANN 2009, p. 79] as well as in requirements management [STEINMEIER 1999, p. 127], applications for the synthesis and management of product architectures are also known.

BRAUN & DEUBZER for example show a combination of the ΣDSM and the consistency matrix for a new variant management approach. The adding up of matrices within the models, each representing the consistency matrix of one variant of the product portfolio, composes the ΣDSM. Within the ΣDSM, the introduced variants, as well as further possible variants based on that information, can be identified through the application of a clustering algorithm. This algorithm helps to identify each variant within the portfolio by pointing out completely interconnected clusters [BRAUN & DEUBZER 2007].

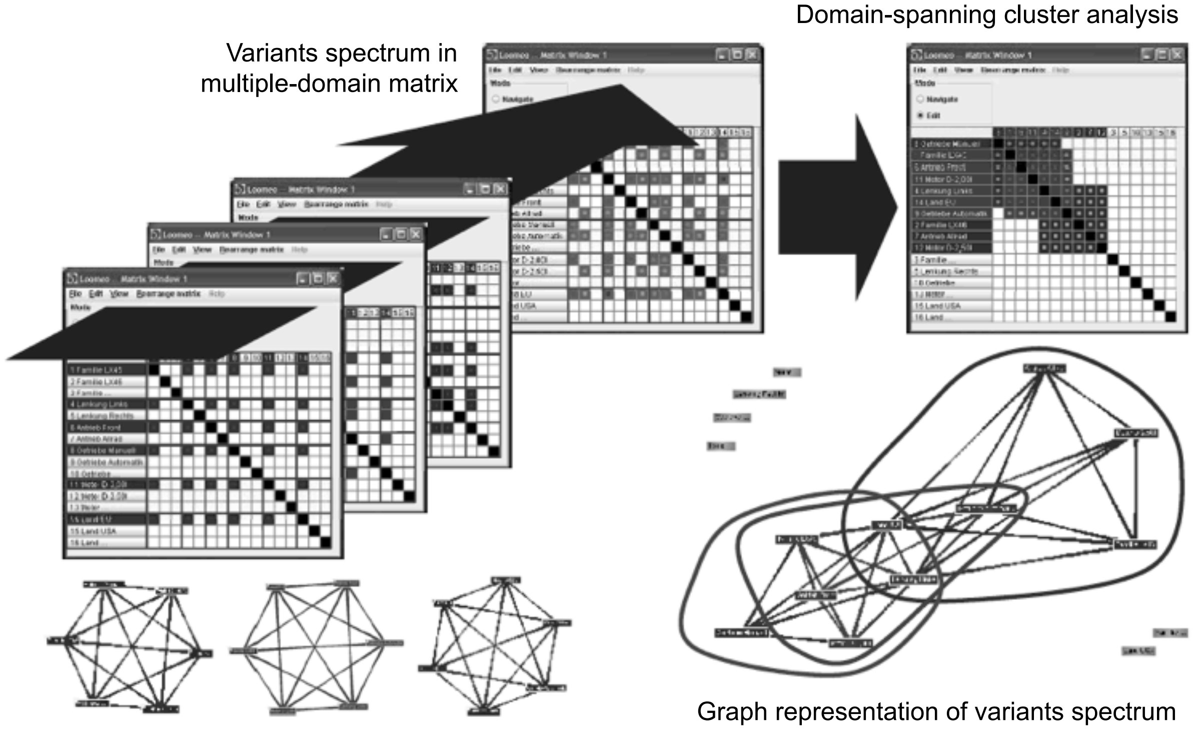

Domain-spanning cluster analysis
Variants spectrum in multiple-domain matrix
Graph representation of variants spectrum

*Figure 5-21 New variant management approach [*BRAUN & DEUBZER *2007]*

HELLENBRAND & LINDEMANN introduce a similar approach, applying the consistency matrix and clustering algorithm as well, but for design synthesis. The consistency matrix is generated through the transformation of a morphological chart of four functions and 24 partial solutions into a consistency matrix. During that process, information is added regarding which partial solutions can be combined and which cannot. As a result, the consistency matrix, similar to the variant management approach previously discussed, shows a depiction of the comprehensive solutions space, instead of a collection of discrete solutions, The clustering algorithm helps in identifying the completely interconnected clusters of all four functions, which represent possible solutions [HELLENBRAND & LINDEMANN 2008].

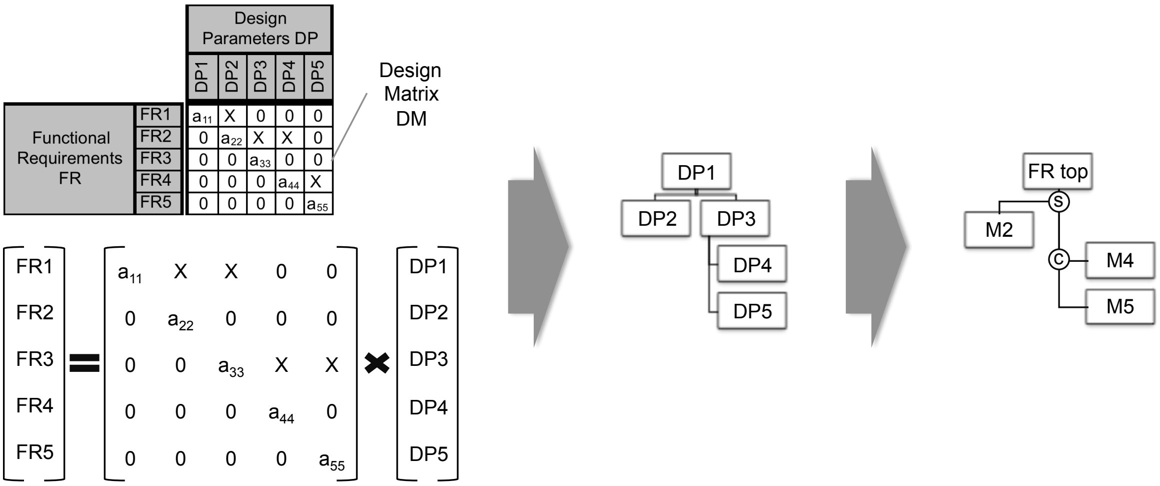


Figure 5-22 Morphological chart with one combined solution highlighted and depicted in matrix notation (MDM consistency matrix) [compare HELLENBRAND *&* LINDEMANN *2008]*

Due to the extent of constrictions and strictly mathematical description, **axiomatic design theory (ADT)** can be seen as an outstanding approach [SUH 1990, SUH 2001]. Though the application for product design, especially new product design, is discussed critically [e.g. WEBER 2005a], axiomatic design is an often cited and discussed approach. Its application exceeds the field of product design, and is also used for organizational means [e.g. LENZ & CHOCHRAN 2000].

Axiomatic design aims at the mathematical transformation of customer needs into functional requirements (what has to be achieved), which are then translated into design parameters (how to achieve it) and later into process variables for production. Constraints are introduced, and neither the design parameters nor the process variables are meant to interfere with them[SUH 1998]. Though a precise mapping of the mentioned entities for design synthesis was underlined in the previous sections, as was the potential of matrix- or graph-based modeling techniques, axiomatic design turns out to be too strict in terms of the mathematical framework, the proposed procedure, and included theorems for application.

The core of axiomatic design is the mapping between functional requirements and design parameters, i.e. the coupling of the functional and physical space. As the following figure depicts, the multiplication of the design matrix and the vector of design parameters result in the vector of functional requirements. An inversion of the equation represents the process of design, from requirements to design parameters. The design matrix is required to be square, reflecting the most important axiom of independence. Each functional requirement is represented by a corresponding design parameter. As such, the theory requires unambiguous hierarchies in both the functional and physical spaces. Additionally, these hierarchies have to correspond precisely [SUH 1998, WEBER 2005a].

In practice, the applicability of the axiomatic design theory is only given in cases where these hierarchies and the theorem of independence can be fully satisfied. As was discussed earlier, this is rarely the case for products. In fact, the complex interrelations between requirements, functions and physical components almost never represent one to one mappings.

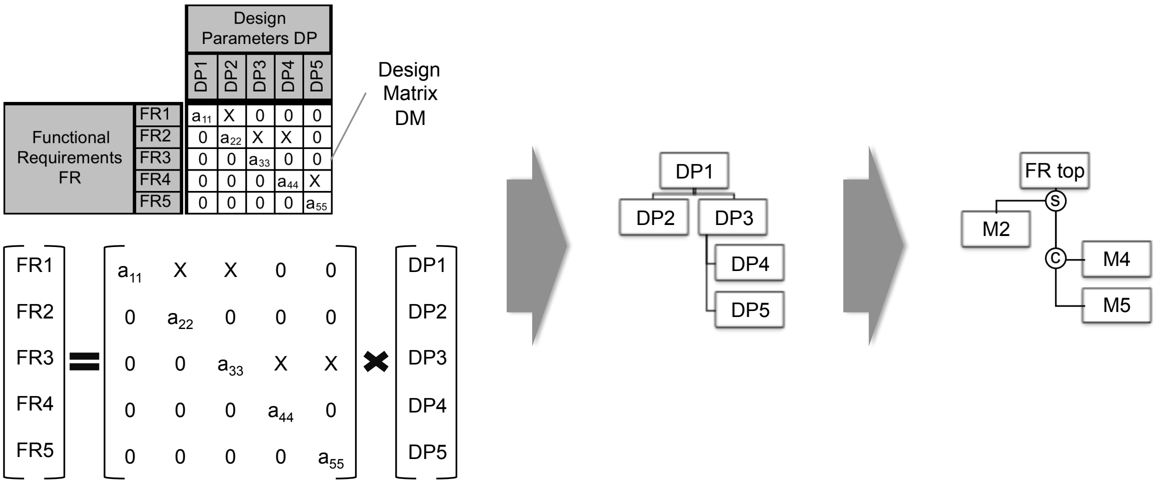


Figure 5-23 Relation of functional space to physical space and resulting modularized architecture, according to the axiomatic design theory [Suh 1998]

Furthermore, the product needs to be decomposable into hierarchies, which follow the “is part of” type (summation, marked “s” in the graph to the right). The design parameters of level three, in the given example DP4 and DP5, together form the design parameter DP3, which itself does not require any further consideration, so that in the resulting architecture the modules M4 and M5 are considered to fulfill module M3 in the control sequence (marked “c” in the graph). Consequently, the superordinate functional requirement is fulfilled by the sum of the subordinate functional requirements, according to the independence axiom [Suh 1998].

While it is worth striving for the fulfillment of the independence theorem, it cannot be satisfied in many cases, reducing the benefit of the theory for highly complex products, which are characterized by the numerous and various interdependencies between entities.

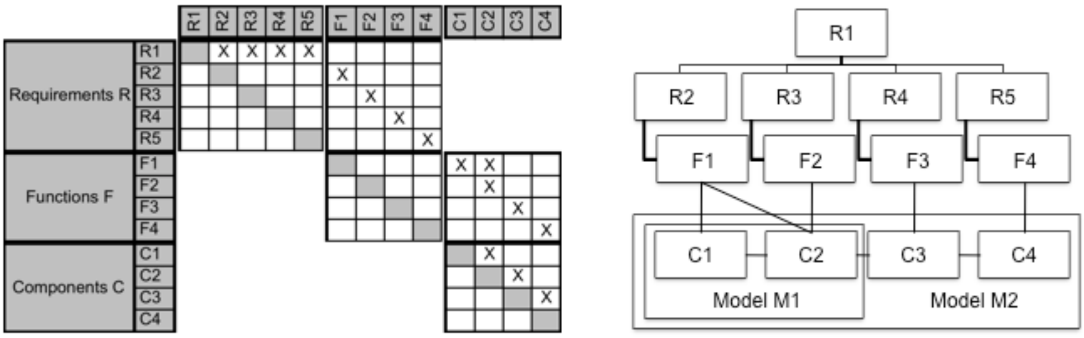
As a last critical remark, the system entities must be known to apply the theory. The application during the early phases of design, when not all entities and relations are known, is impossible, thus narrowing the suitability for synthesis in early phases of design [Weber 2005a].

Some authors, aware of the potential and shortcomings of axiomatic design, couple its application with other methods for product redesign. Guenov & Barker propose an approach for the redesign of products using axiomatic design theory in combination with DSM, supporting the iterative decomposition-integration process during redesign. The different levels of detail are still unsolved in this approach [Guenov & Barker 2005]. Similarly, Mann outlines axiomatic design potential analysis capabilities, yet compares and couples the theory with the TRIZ approach for design synthesis. In the use case example, a mixer tab system is considered as well as redesigning problem [Mann 2002].

For **synthesis in conceptual design**, Kusiak promotes a matrix-based approach similar to that of axiomatic design [Kusiak 1999, pp. 223ff]. He proposes the mapping of the decomposed requirements to functions as a first step [KUSIAK 1999, pp. 206-219], followed by the assignment of components to fulfill the functions.

Components, in KUSIAK's approach, are indecomposable entities of the product and interact through *connectors*, which connect the components' *ports*, i.e. the in- and output of components. KUSIAK further defines models as a set of connected components, representing the idea of *modules* in other approaches [KUSIAK 1999, p. 224].

Components are further related to *functions,* in the sense of flow-oriented operators as in PAHL et al., and characterized by attributes and behavior. *Attributes* represent constants, state variables, dependent variables, input and output variables, conditional expressions, and algebraic equations, which all describe the components. Dynamic *behavior* is described by the generally valid functions "change", "vary", and "channel" [compare PAHL et al. 2007, p. 32], expressible by mathematical means [KUSIAK 1999, pp. 223-227].



		R1	R2	R3	R4	R5	F1	F2	F3	F4	C1	C2	C3	C4
Requirements R	R1		X	X	X	X								
	R2						X							
	R3							X						
	R4								X					
	R5									X				
Functions F	F1										X	X		
	F2											X		
	F3												X	
	F4													X
Components C	C1											X		
	C2												X	
	C3													X
	C4													

Figure 5-24 Synthesis in conceptual design based on the decomposition of requirements and functions [according to KUSIAK 1999, pp. 201ff.]

In contrast to axiomatic design, the approach depicted above allows for a less restrictive modeling and synthesis, due to the lack of theorems such as the independence theorem. As a result, complex products can be modeled, reflecting the ambiguous relationships between functions and components. However, the approach is still largely based on existing solutions, i.e. building blocks in the component domain. In contrast to axiomatic design, the approach presented is more applicable for the early phases in design.

Concluding the chapter on matrix-based methods for design synthesis, a few statements can be presented for the evaluation of the methods presented. First, the approaches are based on systematic procedures, and are thus different from conventional and creativity-supporting techniques (chapters 5.5.3 and 5.5.4); this is comparable with to systematic approaches discussed in chapter 5.5.5. The advantages of systematic procedures thus apply here as well.

Furthermore, matrix-based approaches impressively pointed out the relevance of – and suitability for – the iterative character of the processes of analysis and synthesis. The entities of the architecture considered include requirements and functions as basis above all, as well as components and constraints relevant for different domains.

Considering the scope of the approaches, it turns out that the suitability for product architecture synthesis and management is strongly underlined by the examples and procedures. The product architecture is in all cases within the focus of considerations, i.e. the combination of existing partial solutions and effects.

Based on that, the applications consider redesign problems more often than new product design, which can be considered the common situation in industry. Depending on the level of detail, even the definition of partial solutions can be triggered if building blocks are missing.

5.5.7 Computational synthesis and support

Computational synthesis has become a widespread field in recent years. Many publications are available, striving for automatism and support, e.g. by systematically showing to the designer new solutions and possibilities, mostly based on known components or working principles. The synthesis supported by automated means discussed in the following section is that of system composition, rather than detail design, for which computer-aided support exists, such as simulation and modeling applications [e.g. KORNMEIER & RUDOLPH 2006].[73]

Computational synthesis evolved with expert systems in the 1950s, designing e.g. electric motors or generators [CAGAN et al. 2005]. Later on, computational synthesis was applied to combine design parameters and evaluate the outcome, e.g. in CAD systems [see e.g. FIGEL 1988]. The optimization of static structures evolved in the following years (see examples given by different authors in [ADELI 1994]), while the application of genetic algorithms was carried out for production and transportation means, the optimization of scheduling and sequencing problems, or layout problems [CHENG 1997, pp. 133ff.]. Recently, and more relevant for the definition of product architectures, genetic algorithms were applied to depict the solution space and optimize outcomes and single candidates thereof [CHENG 1997, pp. 16 ff.], or support the development and production of variant-rich products [e.g. SHEA et al. 2010].

From the perspective of users in industry, computational synthesis was considered to be of little benefit until 2002, while activities in research suggested a rapid development of the field [WOOD & GREER 2001, p. 220]. More recent reviews point out practical applications in engineering, which are yet strongly specified but still cannot be generalized for the application of unqualified users, i.e. users who were not involved in developing the method and are not knowledgeable about the details and difficulties of the algorithm. As a result, users to present day are primarily researchers [CAGAN et al. 2005].

CAGAN et al. established a generic process for computational design synthesis, as depicted in the following figure [CAGAN et al. 2005, compare TOMIYAMA & SCHOTBORGH 2007 and HELMS & SHEA 2010]. The steps of solution generation, evaluation, and guidance therein describe the iterative design process. As a prerequisite, the problem description, objectives, and constraints – in short the requirements – are to be formalized and a suitable system representation generated.

[73] Recent overviews on design synthesis with many state-of-the-art contributions are provided by CHAKRABARTI and, with even closer focus on automated measures, ANTONSSON & CAGAN. For a brief overview see CAGAN et al. [CHAKRABARTI 2002, ANTONSSON & CAGAN 2001, CAGAN et al. 2005].

The following sections discuss the steps "representation" and "generation" of the generic procedure to structure the state of the art. Thereby, different types of computational synthesis, as well as certain approaches, are considered in detail. The discussion targets the entities of the product architecture considered in the approaches, as well as the models used, purpose and outcome. The focus in general is placed on the approaches for product architecture synthesis, rather than on methods for the optimization of product geometry or energetic behavior. The step "evaluation" is extensively discussed in chapter 5.6, while the step "guidance", which represents the verification of the computational synthesis system and the improvement of the method by means of e.g. machine learning and artificial intelligence [CAGAN et al. 2005], is not closer considered, due to its uniqueness in computational synthesis.

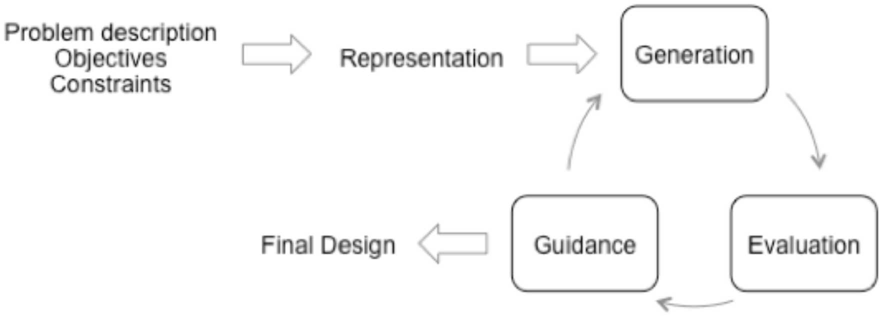


Figure 5-25 Process framework for computational design synthesis [CAGAN et al. 2005]

Representation

The representation includes not only the modeling of the entities available for solution generation, but also the rules by which the product architectures are generated [CAGAN et al. 2005]. The models and related entities are particularly relevant for the discussion in the context of this work. The rules for architecture generation are applicable to the interdependencies of the entities discussed.

Representation and generation are closely interrelated, as the level of detail and focus of the synthesis are defined by the representation, and required represented entities are defined by the generation [CAGAN et al. 2005].

Functional structures are fundamental for the system representation for computational design synthesis [CAGAN et al. 2005], extensively discussed in chapter 5.5.2. Due to their importance for computational synthesis especially, the function-behavior-structure system (FBS-system) and variations thereof are considered in greater detail in the following sections [e.g. GERO 1990, GERO & KANNENGIESSER 2004]. To give an overview, WOOD & GREER provide a classification of function-based approaches, which are all founded on functional considerations, differentiating between the synthesis of dynamic systems, agent-based approaches and catalogue design methods [WOOD & GREER 2001, p. 181].

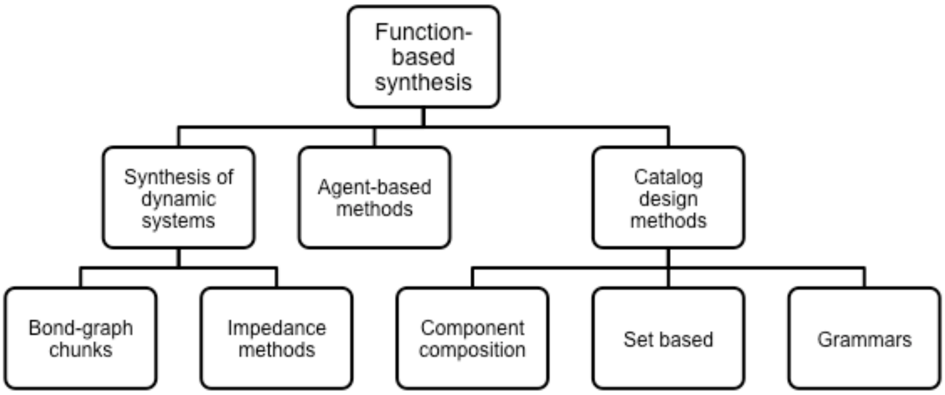


Figure 5-26 Function-based synthesis approaches – overview [WOOD & GREER 2001, p. 181]

For the synthesis of dynamic systems, both **bond-graphs** and impedance methods strongly rely on the consideration of power transformation and thus form – from the perception of function – a subset of the functional perception proposed by PAHL et al. [CAGAN et al. 2005, PAHL et al. 2007, pp. 169ff.]. The bond-graph formalism consists of a set of elements with single or multiple ports, each of them providing a power-flow with certain parameters, all together representing electric circuits, mechanical systems, hydrodynamic, and thermal systems [CAGAN et al. 2005, DAMIĆ & MONTGOMERY 2003, pp. 24-25]. The following figure shows the set of elements and bond-graph (port- and internal) variables.

Table 5-3: Bond-graph variables – overview [DAMIĆ & MONTGOMERY 2003, pp. 7-8]

Domain	Effort	Flow	Momentum	Displacement
Mechanical Translation	Force	Velocity	Momentum	Displacement
Mechanical Rotation	Torque	Angular Velocity	Angular Momentum	Angle
Electrical	Voltage	Current	Flux Linkage	Charge
Hydraulic	Pressure	Volume Flow Rate	Pressure Momentum	Volume
Thermal	Temperature	Heat Flow	--	Heat Energy

As introduced, the **function-behavior-structure system** (FBS-system) [see UMEDA et al. 1990, compare GERO 1990, UMEDA & TOMIYAMA 1995, JIAO & TSENG 1999] is applied as part of the representation through many methods in computational synthesis. Between the functional level and the structural level within the system, i.e. the physical or component level, the behavior-level is introduced, which corresponds with the working model of PONN & LINDEMANN or the level of working interrelationships of PAHL et al. [PONN & LINDEMANN 2008, p. 24, PAHL et al. 2007, pp. 38-41]. In the common understanding of FBS-systems, function describes the purpose of the artifact or system, while the behavior depicts the way that the artifact or system achieves its functions. Structure inherits the artifacts of the system and their interrelations [WANG et al. 2007].

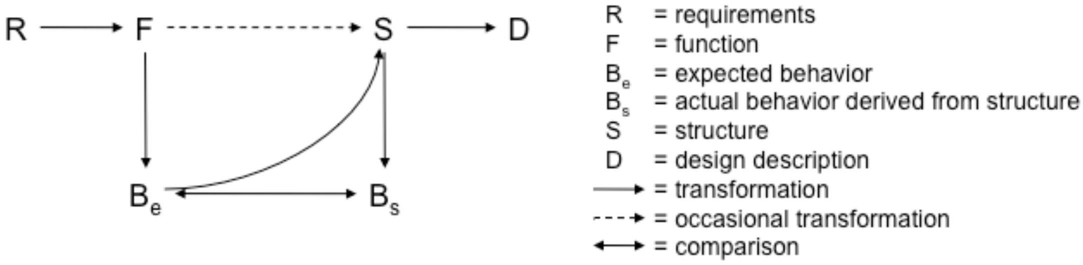


Figure 5-27 The activity of synthesis in the FBS-system [Gero 1990]

Gero shows that the activities in design (formulation, synthesis, analysis, evaluation, reformulation, design description) can be depicted with the FBS-model. The above figure gives an example depiction of the process of synthesis, in which requirements (R) are transformed into function (F), function into expected behavior (B_e), and expected behavior into structure (S). The expected behavior is compared with the actual behavior (B_s) derived from the defined structure, until the design description (D) is reached, based on satisfying results of the comparison. Only “occasionally” can functions be transferred directly into a structure. This is the case for catalogue design problems, for which the solutions already exist [Gero 1990].

The following development of the FBS-system led in two directions. First, users added levels to the FBS-model, such as different functional levels or the level of state [compare e.g. Camelo et al. 2007, Hoisl et al. 2008, Zha & Liu 2005], while the structure of the model itself was revised [compare Gero & Kannengiesser 2004, Wang et al. 2007], separating the expected artifact knowledge space from the interpreted and the working artifact knowledge space for all levels. The former model, separating only two different behavioral spaces (expected and actual or working level), was thus expanded to more thoroughly depict design and design activities.

The following figure seeks to grasp the three different artifact knowledge spaces (expected, interpreted, working), as well as the seven fundamental artifact knowledge elements (F_e, F_i, B_e, B_i, B_w, S_e, and S_w). Functions are then deduced from requirements (R), which are based on customer or developer motivation (M). As a result of design, a design description is provided (D) [Wang et al. 2007].

An example of the synthesis procedure is composed of the following steps, based on the visualized causal relationships: after requirements are derived from expressed needs or desires as motivation for the design, expected functions and behavior are deduced. Synthesis may provide an expected structure. Embodying the expected structure in the working space results in working behavior and working functions of the system. Designers are then able to observe the behavior and function of the system, resulting in interpreted function and behavior. These are compared with the expected function and behavior, resulting in changes to requirements, if needed. Otherwise, the design description can be derived from the structure in the working space. The self-reflective arrows in the expected artifact knowledge space represent possible decompositions of the respective artifact knowledge elements [Wang et al. 2007]. The steps of the procedure differ, depending on the respective activity and the author's perception [compare e.g. DORST & VERMAAS 2005].

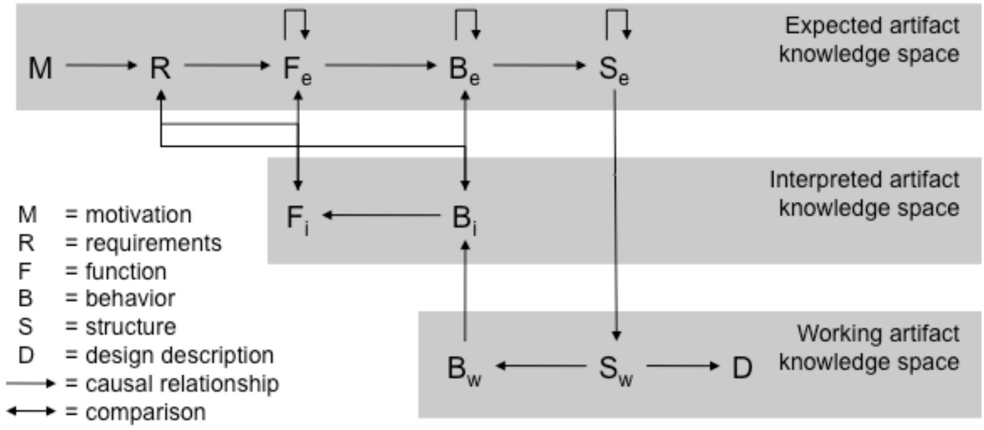


Figure 5-28 The activity of synthesis in differentiated spaces of the FBS-model [WANG et al. 2007]

Apart from the discussed revision of the model, authors add levels to the model to suit the occurring challenges. For example, ZHA & LIU seek an expansion of the information modeling of micro-electromechanical systems. The proposed levels of the model are therefore effects, principles, states, and the environment besides the function-behavior-structure system [ZHA & LIU 2005].

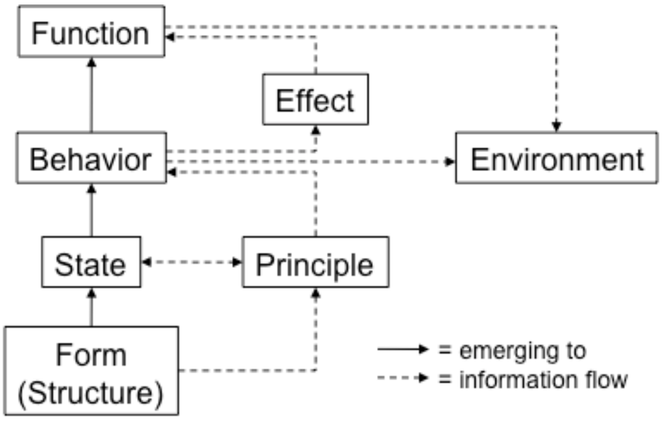


Figure 5-29 Expanded FBS-model [ZHA & LIU 2005]

The behavior-state coupling represents the decomposition of functions into operation and states, according to EHRLENSPIEL, as does the addition of effects and principles to realize function, behavior, and states [EHRLENSPIEL 2009, pp. 390-400]. Structure, in that case, equals form in the model. Though the representation of form is possible in abstract node-edge (i.e. structure) representations, it is not the only possible representation, resulting from an automated synthesis procedure [CAGAN et al. 2005].

ZHA & LIU understand components as essential artifacts of the product, which in hierarchical form compose the product. Artifacts are represented in the model through the aggregation of function, form, and behavior. While function depicts what the artifact is supposed to do, form stands for the design solution, i.e. geometry, spatial description, and its structure, as well as on more detailed level the features (function and form) and attributes (specifications of requirements). The behavior shows how the function is fulfilled. Each class of entities can be decomposed into "part of" hierarchies, according to ZHA & LIU. The most important relationships are the mapping of requirements and specifications, as well as constraints applying to any of the mentioned entities, especially function and form [ZHA & LIU 2005]. KLEIN MAYER et al. support this understanding, defining the fulfillment of functions by technology, realized through physical principles, each characterized by technical parameters and their distribution for the respective effect [KLEIN MEYER et al. 2007].

CAMELO et al. seek a differentiation of functions for conceptual synthesis, separating technical functions (action function) from user functions (purpose function) [CAMELO et al. 2007]. The differentiation thus poses a decomposition of functions as an abstract means.

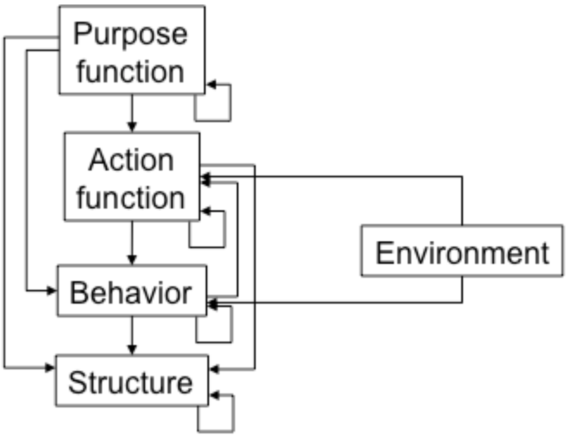


Figure 5-30 Expanded FBS-model [CAMELO et al. 2007]

HOISL et al. introduce a performance model as the linkage between the FBS-system and its requirements, with the goal of mapping of the performance of generated solutions to the requirements [HOISL et al. 2008].

Functional considerations, especially the concept of the FBS-system, inherit great potential for different use cases in design, a few of which are mentioned here. In synthesis terms, not only the design of a product [e.g. by analogy, see QIAN & GERO 1996], but also the definition of product families, can be supported, e.g. by selecting technologies for product families based on the functional representation of different systems [COATANEA et al. 2008]. The analysis of systems can be supported based on the abstract representation, for assessing the degree of novelty of solutions, for example [SARKAR & CHAKRABARTI 2007], or identifying potentially occurring problems during the design of mechatronic systems, in which the functionality of each discipline is usually analyzed separately and in discipline-specific models [D'AMELIO & TOMIYAMA 2007]. Some authors aim for the integration of different functional or port taxonomies based on the concept of FBS-systems as common ground [see e.g. CAO et al. 2008 or OOKUBO et al. 2007].

While the representation of functions and the FBS-system provides the basis for synthesis, the **representation of form** is considered to visualize the outcome of synthesis. As a result of synthesis, depending on the computational synthesis approach, different representations can be considered, ranging from geometrical representations to graph-based node-edge representations of the system structure [Cagan et al. 2005, compare Helms & Shea 2010].

Depending on the applied method of synthesis, the following representations come into question additionally: vector-based optimization, graph structures, and shape and graph grammars [Cagan et al. 2005].

Generation

The extensive discussion of representation methods for synthesis pointed out the importance of representation in the context. Nevertheless, the methods and approaches for synthesis are large in both number and diversity. While the overview by Wood & Greer provides a reasonable classification of methods [Wood & Greer 2001, p. 181], Tomiyama & Schotborgh point out yet another classification [Tomiyama & Schotborgh 2007]. To provide an overview, the following sections discuss different principles of solution finding, as described by Cagan et al., i.e. optimization, search trees, and agents [Cagan et al. 2005], while expert systems and digital solution libraries are added, representing knowledge databases rather than synthesis approaches. Finally, grammar-based approaches are discussed as a powerful means of application of the methods outlined above.

Table 5-4: Synthesis methods – overview [as in Tomiyama & Schotborgh 2007]

Database
Random selection
Database lookup
Database modification rules
Case-based reasoning
Parametric modification
Generation and testing

Generative mechanisms
Backward reasoning
Abduction
Grammar rules
Computational models
Constraint solving
C-K theory's operations

Optimization is a frequent approach to generating novel solutions. In most cases, they seek the optimum solution to a given problem, based on a specified search direction. In contrast, direct search methods explore the solution space, comparing solutions generated based on randomness and selecting the next appropriate one. The use of genetic algorithms is very frequent in the latter case, to enable random explorations of the solutions space in an exhaustive fashion [Cagan et al. 2005, compare also the concept of modification-based design as in Tomiyama & Schotborgh 2007].

Search trees picture design as a sequence of decisions [compare Barton & Love 2000], thus defining solutions based on a chain of decisions along the search tree, while the complete search tree, in theory, depicts all possible solutions. The potential of the search tree unfolds when navigating through making decisions and evaluating the state of the design. The downside, however is that, in contrast to optimization methods, one cannot navigate and compare horizontally to neighboring solutions, but is required to navigate back up in the tree to reverse decisions if demanded by evaluation. After the return to a higher level within the search tree, the detailing of the solution downwards is necessary [CAGAN et al. 2005].

Design **agents**, as a last principle of synthesis, reflect the ability of humans to make decisions, which are substituted by the computational approach. A number of agents might be used to represent different foci of design, while their collaboration can be considered deterministic, random, or stochastic. The process is oriented on the probability of success, based on previously identified designs [CAGAN et al. 2005].

Expert systems strive for the depiction of high level knowledge extracted from experts and stored. Expert systems are mostly applied for the analysis of systems e.g. in medicine. Expert systems are not solely a database of knowledge, but inherit algorithms for reasoning based on the depicted knowledge [JACKSON 1999]. For the presented work, expert systems are not considered more extensively, since the application to the area of engineering design is uncommon and, as such, lacks information on reasonable models and procedures for product architectures.

As in expert systems, **digital solution libraries**, as opposed to paper catalogues [see e.g. ROTH 2001] seek for the identification of solutions in a known solution space. Highly relevant as well as critical to both types is the process of solution finding aided by appropriate logical semantic or automated mechanisms. The logical interdependencies between the entities of the digital library are the foundation for the provision of mechanisms for solution finding. To provide an example, the concept of a digital solution library is given in the following figure [SANDER 2001, p. 98].

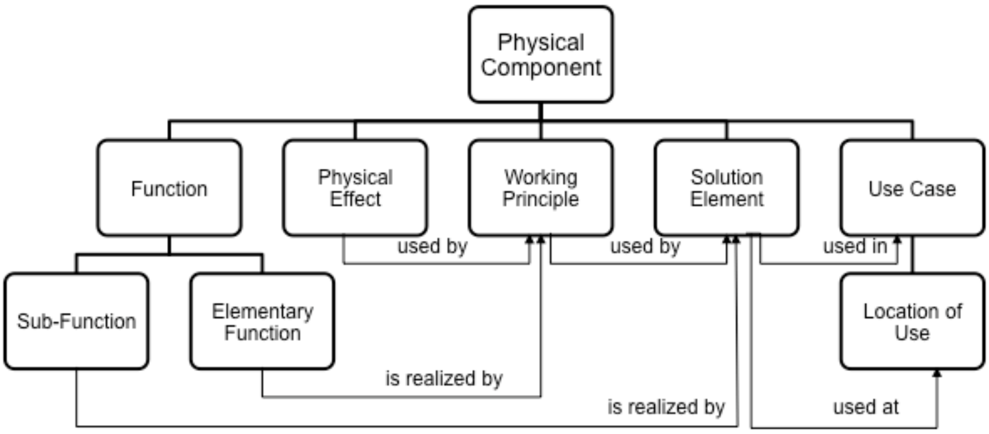


Figure 5-31 Structure of a digital solution library (example adapted from [SANDER 2001, p. 98]

The concept of **grammar-based approaches** makes use of the linguistic understanding of language by providing the available elements in a model and rules to combine those elements, as well as appropriate sequences of the rules to achieve desired goals. Successful examples of application are available in architecture, as well as in engineering. A differentiation is made between graph-grammars, which are generating a graph representation of the structure, and shape-grammars, modifying and depicting the actual form of the system [CAGAN 2001]. The combination is possible, as well [STARLING & SHEA 2005]. The complexity of systems differs, in average remaining low (truss structures, coffee makers) as the methods are lacking a general applicability to different existing problems [CAGAN 2001].

Concluding the section on computational synthesis and support, insight was gained regarding the underlying goals, models, procedures and capabilities of automated synthesis. The dominating perception of functions as a backbone in particular, and the function-behavior-structure models based on that perception, are frequently used for structure or architecture synthesis. As such, important insights were gained about the required entities and procedures. The measures on solution generation discussed provided insight about possibilities for synthesizing architectures based on the provided entities and their model.

5.5.8 Conclusion

System integration as the act of product architecture synthesis requires the following tasks to be conducted successfully [TOMIYAMA & SCHOTBORGH 2007]:

- Designing the product architecture
- Coordinating the mono-disciplinary design and engineering processes
- Integrating the mono-disciplinary design and engineering processes to a comprehensive design at the end of the process

Mirroring the discussed methods against these tasks, not all emerge as feasible for the synthesis of product architectures. Conventional methods, as well as creativity-supporting techniques, are most suitable for mono-disciplinary design processes of subsystems and components. Systematic approaches allow for both mono-disciplinary activities and integrating the processes in terms of combining results of different disciplines. Matrix-based approaches, as well as computational synthesis (as discussed here), mainly intend to design the product architecture itself, based on the results of mono-disciplinary design results.

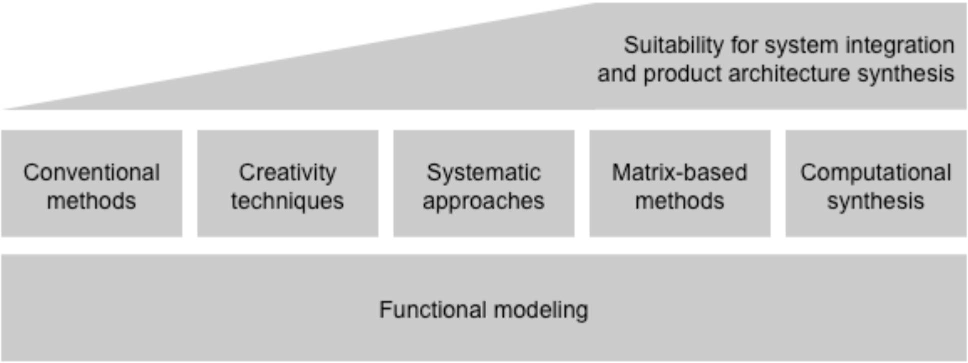


Figure 5-32 Method evaluation for product architecture synthesis

Though the numerous methods have shown potential, challenges still remain, one of which being the still remaining **gap between design synthesis and analysis**. Matrix-based methods and computational methods show the greatest potential for narrowing this gap, since evaluation means are an integral part of the approaches. As the overview has shown, numerous methods exist that are supporting either the synthesis or the analysis. The formalized exchange between analysis and synthesis methods, e.g. to enable the reduction of solutions during synthesis, does exist for the evaluation of solutions within the computational approaches, but lacks generic grounds to interrelate the different existing methods. To present day, the evaluation differs between methods and use cases.

As a second downside, existing methods are **either supportive of system integration or the design of novel partial solutions**. For successful product architecture management, the two areas need to be integrated more closely to point out needs for novel partial solutions, as well as potential for novel integration approaches.

Finally, a sufficient **level of abstraction** is required when applying a new approach successfully, allowing for the definition of novel solutions, as well as the enhancement of existing products.

5.6 Concept and properties evaluation

The evaluation of concepts, properties of concepts, and products marks an important step in the procedure of design in general [ULRICH & EPPINGER 2003, pp. 123-208, PAHL et al. 2007, pp. 106-124], as well as computational synthesis [CAGAN et al. 2005]. Even more important, the process of design can be understood not only as information processing, but also as a sequence of decisions based on the given and generated information [BARTON & LOVE 2000, SIMMONS 2008, p. 18].

The following paragraphs discuss the character of decision-making in design and the different existing methods, which come into consideration for product architecture evaluation. Concluding the chapter, the discussion turns to examples of properties, which frequently add to the decision-making process during product architecture management, as well as accordingly specialized methods.

5.6.1 Decision-making in product architecture management

Since the design process is essentially a chain of decisions [BARTON & LOVE 2000, SIMMONS 2008, p. 18], the importance of decision-making at the various steps of design and development in general is obvious and has been researched [KRISHNAN & ULRICH 2001]. During design, essentially every concretization from level to level, e.g. requirements to functions to working principles etc., is lead to by a decision.

A comprehensive overview of product development decisions, classified according to the four categories of *concept development*, *supply-chain design*, *product design*, and *production ramp-up and launch* is provided by KRISHNAN & ULRICH for the purpose of structuring research in product development [KRISHNAN & ULRICH 2001].

If one considers product architecture management as the analysis and synthesis of product architectures, as well as the maintenance of product architecture portfolios, the relevance of decision-making is greatest for synthesis. As such, the focus of the following considerations is the ability to support product architecture synthesis. Of course, important decisions are made during analysis (compare "selectivity" in chapter 5.4.2) and maintenance (compare "variant management" in chapter 5.7.1), which will also be recognized.

In general, the results of systematic decision processes in product development can be summed up as [ULRICH & EPPINGER 2003, pp. 128-129]:

- Customer-focused product
- Competitive design
- Better product-process coordination
- Reduced time to introduction
- Effective group decision-making
- Documentation of the decision process

In this section, the basic understanding of decision-making is developed to classify decisions in general, and decisions in product architecture management in particular. Grounded on that understanding, evaluation and decision-making methods in the following paragraphs can be systematically compared and assessed, depending on their suitability for the means in product architecture management.

A decision in design can be characterized according to SIMMONS [SIMMONS 2008, p. 19]:

- A decision occurs in a **controllable situation** with **multiple alternatives**.
- A decision during design **separates the solution space**, i.e. eliminates classes of solutions.
- By making a decision, the decision maker expects to **achieve benefit**.

While SIMMONS differentiates decisions roughly into **programmed decisions** (routine, well-defined and precisely modeled, not novel) and **non-programmed decisions** (non-routine, weakly defined, imprecisely modeled, solved by general problem solving methods), HATAMURA offers a differentiation by type and elaborateness of the decision. The resulting types of decisions are "**go or no-go**", "**single selection**", or "**structured decisions**", while the elaborateness separates **poor** from **rich** routes to decisions [SIMMONS 2008, p. 19, HATAMURA 2006, pp. 2 and 7, compare also EHRLENSPIEL 2009, p. 504].

Generally, it is easy to classify a decision by the criteria given above. In particular, "rich" and "poor" decisions deserve additional explanation. The "richness" refers to the degree of clarity in the path along which decisions are to be taken. A poor route has few alternatives, and thus, if a disturbance occurs in one node of the process, the route disintegrates, and the disturbance propagates until the end. Rich decision routes, on the other hand, spread over the solution space with many branches. Since start and end are connected by multiple paths, the rich decision route is far less vulnerable to single disturbances [HATAMURA 2006, pp. 6-7].

On the downside, it is not desirable to have many paths, for the sake of quantity. Alternative paths are only valuable if they are kept as "active paths", i.e. detailing the possibilities, at least virtually, of using alternate routes if required and building them based on the actual state of knowledge. As a result, alternative routes are to be kept modest, i.e. not too spread out, but flexible and able to react to disturbances [HATAMURA 2006, pp. 6-7]. In brief, the decision-maker is asked to know the alternatives, and keep an overview about the possibilities to react quickly.

The process of decision-making in the sense of applying a systematic evaluation method can be roughly sub-divided into the steps of *identification of criteria*, *rating*, *ranking*, and *selection*. The following figure depicts two more elaborate procedure examples, of which PAHL et al. discuss the concept evaluation of a complete system, while ULRICH & EPPINGER consider the evaluation of separate sub-problems' solutions and the combination to an overall concept. Both approaches are valid, and, as the comparison shows, are similar to one another, despite the different names occasionally used [PAHL et al. 2007, pp. 110ff., ULRICH & EPPINGER 2003, pp. 134ff.]. Additionally, LINDEMANN proposes to add plausibility and sensitivity analysis after the discussed steps, to reflect systematically on the evaluation results [LINDEMANN 2009].

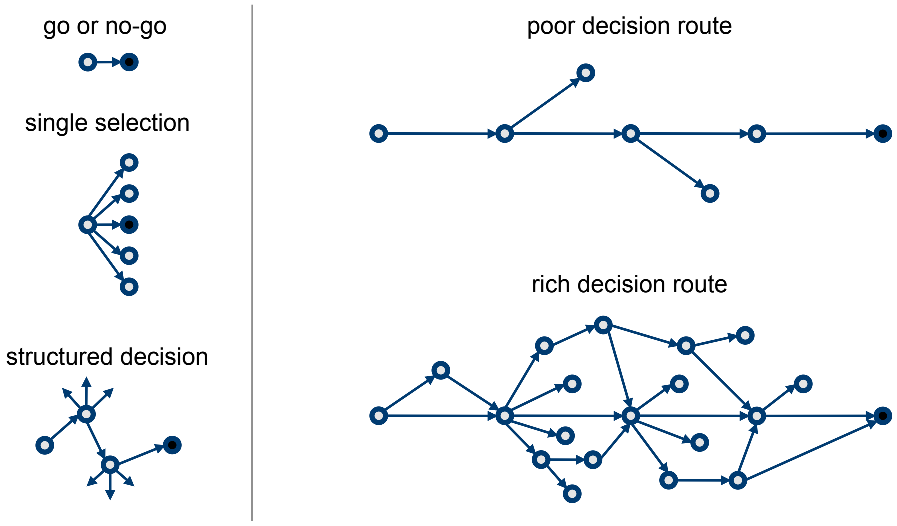


Figure 5-33 Types of decisions (left) and types of decision routes (right) [HATAMURA 2006, pp. 2 and 7]

The process of decision-making can be described in the following steps, which combine the approaches of PAHL et al. and ULRICH & EPPINGER [PAHL et al. 2007, pp. 110ff., ULRICH & EPPINGER 2003, pp. 134ff.]:

- Preparing (identifying evaluation criteria)
- Weighting evaluation criteria
- Compiling parameters
- Assessing values
- Determining overall value (combining and improving)
- Estimating evaluation uncertainties
- Searching for weak spots
- Selecting

Based on the above discussion, it can be argued that decisions during the process of product architecture synthesis are mostly non-programmed decisions, especially during the early phases of design. Decisions are non-routine and the models imprecise and qualitative. The interconnectivity between product architecture entities reflects in complex structured decisions, as opposed to "go or no-go" decisions. If decisions do not result in follow-up problems and thus follow-up decisions, cases of single-selection rarely occur. The routes to decisions are preferably rich, yet hard to handle and maintain. A support of decision-making has to regard the characteristics of decisions in product architecture management.

5.6.2 Methods for decision-making and evaluation

The following sections discuss methods that intend to support decision-making in design. Conventional approaches serve as measures for single selection problems, and form the basis for most of the elaborate measures discussed in detail. Qualitative reasoning and benchmarking are thus considered to be examples for other approaches.

Since evaluation is an integral part of analysis and synthesis, methods already discussed in this work contain aspects of evaluation and decision-making, to at least support the process of decision-making by gathering, structuring, and gaining information. Especially analysis is often related or even equated with evaluation, which is valid for the respective processes as well [BERNARD 1999, p. 51]. SIMMONS, for example, considers the design structure matrix and morphological chart as methods for decision-making for comprehensible reasons [SIMMONS 2008, pp. 31-33]. Methods already discussed in previous chapters will be cross-referenced in the following sections, but explanations will not be repeated. The method overview is structured in methods for single decisions and causality methods,[74] while particular frameworks for evaluation, based on certain criteria such as quality are discussed in chapter 5.6.3. In the beginning, an overview on qualitative reasoning is given, as an adequate introduction for decision-making for product architecture problems.

Qualitative reasoning

Qualitative information, especially in the early phases of design, is often the main source for decision-making. Qualitative reasoning provides the means for these situations, bridging the gap between human perception in these stages and the possibilities of quantitative calculations in later phases. Qualitative reasoning provides possibilities to cope with vague information, typical for human decision-making processes. This tackles the problems of the resolution and the narrowness problems [WERTHNER 1994, pp. 2-4].

The resolution problem describes difficulties that occur when an accurate model of the reality is not available. Numerical solutions require precise data, which is often not available.

[74] EHRLENSPIEL differentiates methods for evaluation into *mental models*, *calculation and key figure comparison*, *simulation*, and *testing*. In this classification, the discussed methods represent the first and second groups, while the latter two can be supportively used, yet require detailed virtual or physical prototypes [EHRLENSPIEL 2009, p. 490].

Methods and models have to incorporate vague and partial information. Since numerical means only allow for a precise answer to precise questions, the narrowness problem describes the need of alternatives and answers to classes of problems [WERTHNER 1994, p. 3].

The models of qualitative reasoning are mathematical, based on graph representations. Graph representation enables easy interpretation, as well as providing the foundation for calculations, i.e. mathematical accessibility [WERTHNER 1994, pp. 117ff.]. The logical interrelations, an important part of qualitative reasoning, are mainly based on ontologies [WERTHNER 1994, pp. 13ff.].

The entities of the models are similar to those considered in the analysis of systems in general and the prepositions in this work, i.e. the representation of values, functional relationships, representation of time, and the representation of structure [WERTHNER 1994, pp. 44-55]. For the context of this work, the concern is less regarding the aspect of solution finding by the means of artificial intelligence aimed at in qualitative reasoning, but rather the underlying principles and logics. Qualitative reasoning represents the decision-making process in a formalized way, serving as a foundation for decision-making and evaluation in the context of product architecture management.

The **tasks of qualitative reasoning** include *diagnosis and monitoring* of devices and their behavior, *design and modeling* of artifacts to deduce expected behavior, *interpretation* of results, *identification of structures* and representation in qualitative models, and the *explanation* of cause-effect relationships [WERTHNER 1994, p. 3].

The principles of qualitative reasoning noteworthy and relevant for the systematic approach to decision-making in general; these are summed up by WERTHNER in the following list [WERTHNER 1994, pp. 8-9]:

- **Compositionality** describes the ability to compose a system on the basis of subsystems and entities, which are interconnected. The behavior of entities can be described by local laws, while the behavior of the overall system, in turn, follows from the behavior of the entities and can be observed by a change of state.
- **Locality** stands for the acknowledgement that occurring effects are always local, and propagate through the system through the interconnection of entities.
- **Function** has to be separated from behavior. Behavior is what a system does, while function is the behavior in the context of certain goals, e.g. a blinking “check engine” light is classified as behavior, while the function fulfilled by that behavior is the indicating of occurring abnormalities to the user.
- **Class-wide assumptions** indicate that identifiable laws for one object of a class of entities are valid for all objects within that class.
- The principle “**no-function-in-structure**” demands a reasonable functional decomposition of the system. Neither the functional description of entities nor their behavior anticipate the function of the overall system.
- A **reduced quantity space** is reached by the standardization of real values into qualitative measures for system analysis. As such, a range between three (e.g. ‘+’, ‘0’, and ‘-‘) and ten values is in most cases sufficient to differentiate and rank entities.

- **Reduced relationships** reflect the reduced quantity space for describing relationships between entities. Complex functional dependencies can be reduced to qualitative value representation, similar to the reduced quantity space.
- Since qualitative reasoning deals with the analysis of discrete situations, a **reduced representation of time** is the result. No information about certain points in time or durations is provided, but the situation before and/or after the considered point in time/event allows for the deduction of conclusions.

The above tasks and principles provide a promising basis for decision-making in the context of this work. This work does not include an exhaustive application of qualitative reasoning; however, this was considered during analysis and applies the principles as guidelines for the academic reasoning.

Methods for single selection

Methods for single selection are similar in that they provide the means for the rating of a number of choices, based on identified evaluation criteria. They can be characterized with reasonably little effort for application and accordingly rough outcomes of evaluation [PONN & LINDEMANN 2008, p. 114]. For the most part, methods of single selection aim towards the evaluation of one "domain", so to speak, i.e. usually the physical product entities are evaluated, while other domains, such as functions or requirements [PONN & LINDEMANN 2008, p. 114], serve as criteria for evaluation.

An evaluation method based on the relative comparison of alternatives is the **pairwise comparison**. Qualitative and quantitative criteria can be used for that approach. The advantage is the quick and easy accessibility of both the method and the results. The criteria can be rated for a more differentiated view. On the downside, the results can only provide a ranking of concepts, and are thus only of reduced value for the decision maker; however, this is sufficient in cases where a quick assessment based on vague quantifiable criteria is needed [PONN & LINDEMANN 2008, pp. 114-115, EHRLENSPIEL 2009, pp. 510-511].

The **score evaluation** provides an absolute measure for evaluation. The criteria are scored for each alternative, while the overall values of alternatives provide a support for decision-making. Absolute values provide an improved evaluation compared with the pairwise comparison. Alternatives are often close in the ranking, while the application of more elaborate evaluation methods should be considered [PONN & LINDEMANN 2008, p. 115, EHRLENSPIEL 2009, p. 511].

A more elaborate evaluation method is, for example, the **weighted score evaluation**, which provides an even larger differentiation by weighting the applied evaluation criteria. In order to sufficiently differentiate the alternatives, the range of scores should be exponentially distributed [PONN & LINDEMANN 2008, p. 115, EHRLENSPIEL 2009, pp. 511-512]

Value benefit analysis is based on a hierarchical decomposition of evaluation criteria. The branches of the hierarchy are then grouped according to classes of criteria, such as cost, technical, criteria, etc. The structured approach makes an evaluation of more complex systems possible, based on the step-by-step procedure provided by the hierarchical decomposition of criteria [PONN & LINDEMANN 2008, p. 115, EHRLENSPIEL 2009, pp. 514-517].

Causality methods

For structured decisions, causality methods are the means for approaching the evaluation problem. They intend to unravel the cause-effect relationships of systems [BERNARD 1999, pp. 64-65]. In the following, causality methods are divided into approaches based on graph representation and matrix-based approaches. Common of all types of causality methods is the perception of decision-making as a sequential process [SIMMONS 2008, p. 34].

Decision trees represent each possible sequence of decisions as a branch of the tree. Areas of application are machine learning or data mining, rather than engineering or management, where tree structures are rather the basis for the definition of rules [HAN & KUMBER 2001]. While simple decision trees are made up solely of decision nodes, more complex approaches include chance nodes [SIMMONS 2008, p. 34]. The end nodes or leaf nodes of each branch represent a "complete" sequence of decisions, e.g. a design concept in the context of product development. In more elaborate approaches, where chance nodes are involved and probabilities play a major role, algorithms support the calculation of leaf node values [SIMMONS 2008, p. 34]. Aside from the positive effect of structuring and documenting decisions, the downsides of decision trees outweigh the benefits. First, the hierarchical structure makes it impossible to consider independent or parallel decisions. The underlying assumption states that every decision is influenced by and based on the previous decision. Second, due to that structure, decision trees become very large for even medium-sized sequences of ten decisions or fewer [SIMMONS 2008, pp. 34-35, HAN & KUMBER 2001, p. 306]. The overview and support in decision-making by intuitive application of the tree is then impossible; only algorithmic procedures can guide the decision-making process.

Decision trees, in most cases, depict the physical representation of product architecture, for example during the application of product benchmarking projects [SABISCH & TINTELNOT 1997, p. 131]. BARTON & LOVE extend the concept of decision trees or **design decision chains** to the impact on the organization, process, etc. [BARTON & LOVE 2000], which is also not unusual in integrated benchmarking processes [SABISCH & TINTELNOT 1997, p. 134].[75]

Networked graphs, in contrast to hierarchical graphs, allow for the compensation of the downsides of hierarchical trees; for decision-making in particular, these are the size of the diagram, as well as the strictly sequential decision processes.

Influence diagrams, which represent the causal interdependencies between decisions and allow for a more compact depiction of the information flow between decisions, are representative of networked graph representation in decision-making. Additionally, there is clear visualization of which decisions are independent from one another, and which decisions are directly or indirectly influenced by other decisions.[76] The advantages of influence

[75] Benchmarking as powerful methodology for the systematic comparison of products and/or processes is not intensively discussed in this work. The principles and methods applied during benchmarking correlate to those discussed here. For an overview and possible application procedures [see e.g. SABISCH & TINTELNOT 1997].

[76] To clarify the discussion, SIMMONS provides both representations of the same problem [SIMMONS 2008, pp. 34-36, according to COVALU & OLIVER 1995].

diagrams show in the accessibility to structural reasoning of the representation (compare chapter 5.4.3), as well as in the depiction of cause-effect relationships and a reduction of required space for representation. Additionally, an optimized decision tree can be derived from the influence diagram [Simmons 2008, pp. 35-36]. As an extension, the more elaborate **sequential decision diagram** includes alternatives in the representation, i.e. it depicts which relations become irrelevant due to a choice made previously [Simmons 2008, pp. 36-38].

The matrix-based approaches, of which Simmons explicitly mentions the **DSM** and **morphological chart** [Simmons 2008, pp. 30-33], while Ulrich & Eppinger propose the **selection matrix** [Ulrich & Eppinger 2003, pp. 134ff.], are equally able to represent a network of decisions. Naturally, for decisions required during synthesis in particular, the approaches discussed in chapters 5.4.3 and 5.5 can be applied to compare different alternatives. The selection matrix, which was not discussed previously, allows for the mapping of concepts to evaluation criteria and represents from the core a (weighted) score evaluation, i.e. ratings are given for each concept and criterion, of which the sum results in an overall rating for each concept [Ulrich & Eppinger 2003, pp. 134ff.].

5.6.3 Criteria for decision-making

While Ulrich & Eppinger state that evaluation criteria are meant to be derived from customer needs [Ulrich & Eppinger 2003, p. 131], Ehrlenspiel mentions interviews, interdisciplinary discussions, documentation and visualization of criteria and requirements analysis [Ehrlenspiel 2009, pp. 505-507]. Pahl et al. provide a checklist of areas to be considered when aiming at the derivation of evaluation criteria, depicted in the following table.

Table 5-5: Areas for the systematic deduction of evaluation criteria [Pahl et al. 2007, p. 193]

Checklist of evaluation criteria
Function
Working principles
Embodiment
Safety
Ergonomics
Production
Quality control
Assembly
Transportation
Operation
Maintenance
Recycling
Cost

Thorough analysis of the table shows that it is essentially the activities along the product lifecycle and typical classes of requirements (such as safety, ergonomics), or Design for X aspects, which are used as classes for evaluation criteria. At the same time, the areas could serve as classes of requirements, for example. In the end, if a thorough and comprehensive requirements analysis was conducted, the evaluation criteria should be easily be derived from the resulting requirements list.

The following section discusses rather unquantifiable and inexplicit means of evaluation criteria, as they are regularly consulted in the area of complex product architectures and firmly based on the systems and structure approach strived for in this work. These criteria are change, quality and the class of structural criteria, and should be considered as elaborate examples, rather than an exhaustive list. With time, quality, and cost being the core issues and challenges in product development [CLARK & FUJIMOTO 1991, p. 70, EHRLENSPIEL et al. 2007, p. 21, LAWSON & KARANDIKAR 1994, WILDEMANN 1999, p. 18], methods for change- and complexity-management aim for the prevention of time-consuming and costly processes by identifying purposeful product architectures and enabling anticipatory decisions in early phases.

Change as evaluation criterion

Change in engineering design can be considered in two ways: either as the deliberate change of the product architecture e.g. to better suit the customer's needs [FRICKE & SCHULZ 2005], which is described as redesign or design for customization [CLARKSON et al. 2004]; or the undesired necessity to conduct changes of product entities or artifacts, which at that point were already formally approved and considered unalterable [DEUBZER et al. 2005]. These changes can affect product specifications, functions, components, etc. during product development or production. Changes, especially of the first type, may occur during the whole lifecycle of the product, which is why the product architecture has to enable changeability [FRICKE & SCHULZ 2005]. ECKERT et al. differentiate those types of change into *initiated change* and *emergent change*. Emergent changes are caused by problems along the lifecycle, according to that definition, while initiated changes are caused rather by innovations and requirements than problems [ECKERT et al. 2004]. FRICKE & SCHULZ introduce aspects of changeability, of which *flexibility*, *agility*, and *adaptability* can be assigned as positive aspects to initiated changes, while *robustness* reflects the protection against emergent changes, for which adaptability might provide a supportive means in the sense of controlling emergent changes [FRICKE & SCHULZ 2005].

This ability to cope with changes can be used as a criterion for the comparison of concept alternatives during synthesis or in the context of variant management or customization projects. A central concept to characterize product change is the change propagation, i.e. the acknowledgement that changes cause further changes [ECKERT et al. 2004]. For the analysis of change propagation in systems, networks depicting change interdependencies, as well as matrix-based analysis methods such as the DSM, are frequently consulted. Based on the analysis, product entities can be classified as follows, according to ECKERT et al. [ECKERT et al. 2004]:

- **Constants** are neither actively nor passively affected by change.
- **Absorbers** are more passively affected by change than they are actively inducing change.
- **Carriers** are balanced concerning their active and passive change behavior.
- **Multipliers** tend to cause more changes than they are able to absorb.
- **Buffers** are considered to be entities encompassing tolerance margins, able to absorb change. Whether buffers emit further change depends on the changes which were already cumulatively absorbed, i.e. if no more changes can be absorbed, propagated changes are inevitable.

Based on the classification of product entities, the predictability of change is improved, which in the following section can be utilized to compare different architectures. For example, a product architecture with more absorbers and buffers is preferred to a solution with more carriers and multipliers [ECKERT et al. 2004]. Use cases indicate differential significance of the respective classes. As such, simulation results based on the likelihood of changes [CLARKSON et al. 2001] indicate that multipliers are more meaningful for change propagation than absorbers [OH et al. 2007]. For further insight, organization as well as processes can be assessed concerning how they impact and cope with change [ECKERT et al. 2004, LUH et al. 2011]. KOH et al. introduce an adapted house of quality for the comparison of concepts, based on a mapping of components to features and attributes [KOH et al. 2009].

To cope with change and characterize product architectures, FRICKE & SCHULZ consult the aspect-principle-correlation matrix, in which principles and aspects of changeability are mapped, pointing to difficulties and goals of system analysis for the comparison of architectures [FRICKE & SCHULZ 2005].

Quality as evaluation criterion

The consideration of quality is of high importance, especially in the early phases of development, since shortcomings in quality, though occurring in the early phases, are identified not until the late phases. Changes in late phases, if possible at all, are then costly [FELGEN 2007, p. 3]. Methods for quality management are numerous [FELGEN 2007, pp. 170-172]. In the following sections, Quality Function Deployment and Failure Mode and Effect Analysis (FMEA) are considered, due to their comprehensiveness for a systems approach and extensiveness in application in industry.

Quality Function Deployment is generally associated with the mapping of customer demands to technical (or engineering) characteristics, allowing for the weighting and targeting of values on both sides [AKAO 1992]. The overall goal is the planning of product functions, in line with the customer's perception of quality-relevant properties [AKAO 1992, p. 15]. As a result, core areas of the product for development efforts are identified, necessary trade-offs pointed out and a comparison with competitors made possible. The core of the method consists of the so-called house of quality, a matrix-based representation of the previously mentioned subject matter. The following figure depicts a reduced version of the house of quality, as introduced by MAURER, showing the core features of the method [MAURER 2007, p. 61]. Other authors highlight and add different features of the method, arranging the house of quality in different manners accordingly, and thus showing the different capabilities and application areas of the approach [compare FELGEN 2007, p. 85, PONN & LINDEMANN 2008, p. 41]. The visualization includes the identification of customer requirements and the ratings of customer importance. Customer requirements are coupled with the technical features of the product, which are characterized by technical evaluation criteria. Technical features are correlated within the roof of the house of quality, identifying whether technical features are supportive of one another or conflicting. Further criteria and attributes allow for the comparison with competitors.

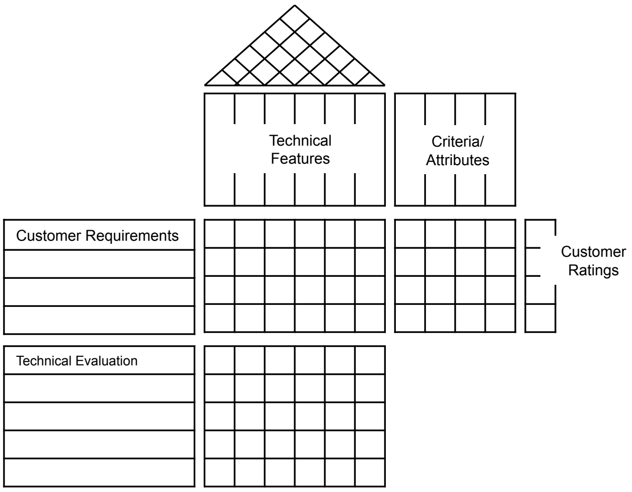


Figure 5-34 QFD House of Quality [compare MAURER 2007, p. 61]

A major challenge of the method is the mapping of customer demands to the technical characteristics, even though literature does not provide a conclusive approach regarding how to obtain that information. As a solution, one approach is to integrate the Kano-model, classifying requirements as must-be, one-dimensional (i.e. represented by a linear additive function to customer satisfaction), and attractive requirements [DE POEL 2007].

Due to the generic representation of the house of quality in matrix-form, it can serve as the foundation for different analytical approaches that can be applied. As an example, VAN DE POEL introduces the correlation between engineering characteristics, the amount of resources necessary to meet certain targets, available budget or overall customer satisfaction etc. as typical variables and calculations of the approach [DE POEL 2007]. Based on the perception of product architecture management, the following chapters will discuss this generic applicability.

Though use cases exist (see e.g. [TSUDA 1997]), a number of difficulties remain regarding the practical application of QFD. VAN DE POEL sums those up as the identification and assignment of customer demands and their impact on overall customer satisfaction. Additionally, he mentions the translation of customer demands to technical characteristics, which can seldom be uniformly conducted [DE POEL 2007].

Failure Mode and Effect Analysis (FMEA) also aims for the elimination of a product's failures and shortcomings. FMEA is thus focused on the dependencies between the technical entities, rather than on customer interaction [MCDERMOTT et al. 2009, p. 3]. This is true for different types of FMEA, of which FELGEN names the system-FMEA, considering modular physical product entities in the product conception phase, design-FMEA for physical components during detail design, and the process-FMEA, considering manufacturing and assembly during production planning [FELGEN 2007, p. 90].

Given the different types of FMEA and its widespread application in industry, the methodology can best be described by a coarse outline of the process and a typical form used for method application [see McDermott et al. 2009, pp. 23ff., Felgen 2007, p. 90].

The first step is the *preparation of the FMEA*, consisting of the allocation of human resources [McDermott et al. 2009, pp. 11ff.], a thorough system and functional analysis (see Means for system analysis and Functional analysis in chapters 5.4 and 5.5.2). The second step is the *risk analysis* on the basis of the functional decomposition. For each function, potential failure-types, -effects and -causes are identified. Networked system analysis (see chapter 5.4.3), depicting causal dependencies can support this process. Maurer & Kesper provide an example for a sophisticated measure of networked system analysis in the context of FMEA, deducting the interdependencies of functions based on FMEA analysis [Maurer & Kesper 2011]. As a third step, *risk evaluation* is conducted, rating the possibility of occurrence of the failure, the impact, and the possibility of detection of the failure. The overall rating is represented by the risk priority figure, resulting from the multiplication of the three figures mentioned above [compare McDermott et al. 2009, pp. 23ff., Felgen 2007, p. 90].

Structural characteristics as evaluation criteria

Based on the discussion of change and quality as evaluation criteria, the importance of the product architecture became evident. Change and quality strongly rely on characteristics, principles, and causalities, which are directly represented by and inherent within the product architecture. For that reason, not all structural characteristics, as summarized in chapter 5.4.3, will be discussed. As an example, the product architecture characteristics relevant for change and quality will be presented in greater depth.

Following the classification of architecture entities according to Eckert et al. [Eckert et al. 2004], suitable structural characteristics can be assigned to the classification, underlining the importance of the product architecture. The structural characteristics allow for a systematic analysis of the overall product architecture and can support the comparison of architectures by the occurrence of the respective characteristics. In the following table, the propagation node type is listed in accordance with the classification of entities, while node characteristics and subsets are assigned to the propagation types.

Table 5-6: Assignment of structural characteristics to node propagation types

Propagation type	Node characteristics	Subset criteria
Constant	Active and passive sum, isolated, end and start node	Feedback loop, hierarchy
Absorber	Active and passive sum, end node	Feedback loop
Carrier	Active and passive sum, transit node	Feedback loop
Multiplier	Active and passive sum, start node	Feedback loop, hierarchy
Buffer	Active and passive sum	Feedback loop

Constants (neither actively nor passively affected) are characterized by a low active and passive sum, since changes are unlikely for that type of node. End and start nodes are likely to be constant (especially start nodes), while isolated nodes are in any case constants in terms of change propagation. Constants are not to be involved in feedback loops or hierarchies, i.e. subsets predestined to cause large change propagation.

Absorbers (passively affected rather than actively) are characterized by a relatively high passive and low active sum. End nodes are potential absorbers, while, on the other hand, absorbers are unlikely to be part of feedback loops.

Mediators of change are *carriers* (balanced active and passive behavior), which inherit a similarly high active and passive sum. Transit nodes are likely to be carriers. Feedback loops may contain carriers.

Multipliers (causing more change than absorbing) of change are characterized by a higher active than passive sum. Start nodes can be classified as multipliers, though not containing input changes. Multipliers may be part of feedback loops and hierarchies of change.

Buffers (encompassing tolerance margins) are difficult to characterize by structural features, since the changes are absorbed to a certain amount. A high passive sum is an indicator for soon-to-be cumulatively absorbed changes, and thus the propagation of changes. The active sum is of lower relevance for the identification of buffers. Buffers in feedback loops are likely to turn into carriers, since feedback loops are an indicator for the consumption of the tolerance margins.

The identification of the respective propagation types by structural characteristics cannot fully be accomplished. However, the analysis of structural characteristics can give hints about the overall change propagation performance of a product architecture and narrow down the candidates for each propagation type. The structural characteristics further help to classify the entities in detail.

The relevance of structural criteria for change can be discussed more thoroughly, based on the principles summarized by FRICKE & SCHULZ, which are explained briefly in this work, while their elaborate description, interaction and examples can be found in the respective source [FRICKE & SCHULZ 2005]. Principles mentioned can be equally defined by structural characteristics.

The propagation types and principles of change discussed above clearly show that the structure, i.e. the product architecture, is one of the main characteristics of change and has the strongest implication on the change behavior of the product.

Similar to the relevance for change, structural properties of the product architecture are equally important for the assessment of a product's quality. Within the approaches of QFD and FMEA, the interdependence of requirements, functions and components is the major factor and backbone of the methods. Although structural characteristics in quality management are highly relevant, they are just as diverse and cannot be assigned in a generic fashion. Since the coupling of entities, such as requirements, functions, and components, as well as the interrelations within the domains, is the crucial aspect, application of structural characteristics is very high, but has not yet been conducted [MAURER 2007, p. 61].

Due to the complexity of quality problems, individual analysis criteria cannot be directly assigned. However, the potential of methods for system analysis can be clarified, if the methods reveal causal relation chains (feed-forward and trace-back analysis) [MAURER 2007, pp. 233 and 237] and critical entities (mine seeking, structural ABC-analysis) [MAURER 2007, pp. 235-236].

5.6.4 Conclusion

Based on the examples of change and quality, the importance of the product architecture and the relevance of structural criteria was discussed. Given this dependency between concrete evaluation criteria, such as change and quality, and structural characteristics, the significance of an overall classification of product architectures is a possibility to assess generic product architecture properties. Although different evaluation criteria are relevant in varied situations, an overall analysis is nevertheless able to compare product architectures, because they reflect on concrete evaluation criteria.

5.7 Downstream activities

The downstream activities discussed in this chapter focus on the perspective of the action system, due to their position at the end of the chapter. Additionally, different aspects of the downstream activities are relevant for the product, i.e. the object system, or pose requirements to be considered in early phases. Although the many downstream activities could be addressed in greater depth, the aspects such as service, production, assembly, recycling etc., are considered under the caption "lifecycle management", which could also be entitled "Design for X". Variant management is discussed as a distinct topic, due to its importance in recent product development projects and the acknowledgement that the design of future products will, in most cases, be embedded in an existing product family or be the start of a product family in the future [FRICKE & SCHULZ 2005]. The existing approaches in variant management, product family design, modular product design etc. will provide an overview, while at the same time pointing to potential that is yet untapped, which could be raised through application of the approach developed in this work.

5.7.1 Coping with variants

The importance of niche markets and the resulting need for variant rich products was extensively discussed in chapter 1.1.1. The deliberate and systematic consideration of product architectures thus cannot be conducted without taking variants into account. The positive or negative results of variant management activities show in the late phases of the product lifecycle, such as manufacturing. For that reason, the coping with variants is discussed under the notion "downstream activities" in this chapter. To avoid misunderstandings, their placement in this chapter is not meant to give the impression that the consideration of variants is irrelevant in the early phases. On the contrary, the groundwork for successful variant management is laid out in the early phases of requirements management, product planning, development, etc.

The following sections discuss the coping with variants, which are necessary to fulfill the arising requirements. First, the definition of variant management is stated, accompanied by an overview of frequent goals of variant management activities. The process of typical variant management projects is elaborated, while the concluding sections introduce an overview of methods and models for the different goals in variant management. The overview is intended to introduce and compare representative methods with the goal of completing the product architecture model and identifying methods suitable in the context of product architecture management.[77]

Definition and goals

Variant management as such means managing complexity caused by the market and customers, which is why it is often seen as a subcategory of complexity management [DEUBZER ET AL. 2008]. For a long time, variant management meant focusing on the most frequently ordered variants; however, niche markets and individualized products gain more importance and stand for profit for the companies at present and in the future [ANDERSON 2008]. Variant management can be defined as the sum of all measures, which are intentionally influencing the range of variants within the company. The intentional effect on variety can be manifested in products or product architectures and processes [PONN & LINDEMANN 2008, p. 231, KUSIAK 2002]. The overall goal of variant management is to influence the complexity, so that the *external complexity* (market and environment) is high, while the *internal complexity* (within the company) is as low as possible [PONN & LINDEMANN 2008, p. 231, RENNER 2007, pp. 22-23, JIAO & TSENG 1999], representing the area of conflict between flexibility and cost.

To complete the variant management paradigm, SIMPSON differentiates platform strategies, based on the chosen clustering of the product portfolio to cope with the internal complexity. Three different strategies can thus be identified: *horizontal leveraging* (extending a platform across different segments), *vertical leveraging* (extending a platform across the range of low-, medium-, and high-priced, -quality or -performance levels), or the combination of both [SIMPSON 2004, see also SIMPSON et al. 2001 and MEYER 1997]. JIAO et al. differentiate between *scalable* or *configurational* product family design [JIAO et al. 2006]. Side effects of modularization include the recycling or retrieval of products [ZHANG & GERSHENSON 2001], or the structuring of processes accordingly to the product modularization [LUH et al. 2011]. Despite the benefits and efforts in variant management, limitations exist, especially when discussing mechanically-dominated products in contrast to, for example, services or highly electronic products [WHITNEY 2004].

Variant management, as well as its diversity, can be characterized by the goals and activities typical of variant management projects and approaches. Different foci in variant management target the product portfolio (cost), processes & organization (effort, time, competences, etc.), development (modularization, reliability) or production (cost of manufacturability, quality, time) as fields of activity [compare DEUBZER et al. 2008, KUSIAK 2002, RENNER 2007, pp. 118-120]. Most benefits of variant management projects are achieved by the modularization of the product, allowing for the reuse of developed modules. As a result, development time and competencies can be purposely appointed (*reduction of effort*), while the manufacturing

[77] An overview of the general nomenclatures and situation is provided by DU et al. [DU et al. 2001] while SIMPSON et al. provide a collection of methods [SIMPSON et al. 2006, see also FIXSON 2007]. An overview of both in German language is provided by RENNER [RENNER 2007].

and assembly processes can be conducted more efficiently and the number of required tools reduced (*reduction of cost*). Since modules can be independently tested, product properties can be improved (e.g. *quality* or *order lead time*) and the combination of modules allows for flexible reactions to customer needs (*increase of flexibility*). Apart from the modularization of the product as the main activity, approaches aim for the monetary aspects of variants by reducing complexity within the product portfolio. In general, the *reduction of complexity* can be achieved by the reduction of number and variety of elements characterizing the product portfolio. Elements can thus be different entities of the product architecture, e.g. requirements, functions, components, variants, etc. Frequent examples are the *harmonization of requirements* and the *elimination of variants* not frequently requested by customers. The following table sums up the discussed goals and activities of variant management, as well as representative examples in literature utilizing the approaches [compare Renner 2007, pp. 118-120, Kusiak 2002]. To complete the picture, the different goals and activities need to be accomplished through the close collaboration of many, usually interdisciplinary, departments within the manufacturing company such as sales, development, manufacturing and assembly, testing, controlling, etc. It is important to note at this point that a successful variant management project has to incorporate interdisciplinary teams of all areas to achieve cost-transparency, purposeful technical solutions, cost-reduction in manufacturing, and, finally, successful products. The importance of each discipline will nevertheless differ, depending on the prioritized goals and chosen activities.

Table 5-7: Typical goals and related activities in variant management

Goal	Activity	Sources (examples)
Reduction of effort (resources) (development time, team structures, competences,...)	**Modularization** (Use and reuse of developed modules, functional modularization**)**	Avak 2007, Kusiak 2002
Reduction of cost (manufacturing, tools, assembly,...)	**Modularization** (Use and reuse of developed modules)	De Lit & Delchambre 2003, Kusiak 2002, Umeda et al. 2005, Williams et al. 2007, Zhang & Gershenson 2001, Park & Simpson 2008
Improvement of quality (or other properties)	**Modularization** (Use and reuse of tested and validated modules)	Kusiak 2002
Increase of flexibility (customer satisfaction through customer worth differentiation)	**Modularization** (use of existing modules, increasing reactivity, high number of possibilities, cost-efficient application, increase in degrees of freedom)	Martin & Ishii 2002, Kusiak 2002, Suh et al. 2007, De Weck 2007, Bongulielmi et al. 2002
Reduction of complexity	**Reduction of number and variety of elements** (requirements, functions, components, technologies,…)	Bongulielmi et al. 2002, Wildemann 1999, Schuh 2005

Optimization of requirements	**Harmonization or differentiation of requirements** (avoidance of functional underperforming and reducing overperforming)	Simpson et al. 2001
Simplification of portfolio	**Elimination of variants** (not requested or rarely requested, exchangeable)	Thevenot & Simpson 2006, Farrell & Simpson 2008

The discussion above allows for a comprehensive differentiation of variant management activities, according to the leveraging strategies, fields of activity and goals. The technical realization and different possibilities will be discussed within the following sections.

Process of variant management

Renner provides a rough yet reasonable outline for a procedure for the development of product families, based on existing products and an existing product portfolio. Given the variety of approaches and different directions of goals, his generic proposition of five steps turns out to be representative and useful in projects with different emphases [Renner 2007, pp. 100-110].[78]

The first step consists of the *prioritization* of goals and matters, identifying which goals are the foci of the variant management project in general. Possibilities are represented for example by the goals, activities and their combinations, discussed in the previous section. Suh et al., as well as Simpson et al., discuss the identification of markets and uncertainties, for example [Suh et al. 2007, Simpson et al. 2001].

Following the prioritization of goals, the *analysis* of functions, requirements, boundary conditions, components, etc. and the complex interrelations is conducted. The main goals of analysis are to achieve transparency about existing solutions and to identify variant drivers, cost structures, etc. Methods for analysis are discussed in chapter 5.4. Suh et al. stress the importance of coupling of variant attributes with market segments and the resulting platform bandwidth [Suh et al. 2007]. Additionally, the modeling during analysis to support the following synthesis and evaluation procedures is to be conducted in this phase [Simpson et al. 2001]. Williams et al. propose the modeling of non-uniform demands to grasp the distribution of requirements in the early phases of design [Williams et al. 2007].

The *synthesis* as the third step of the procedure includes the identification of possible modules, i.e. core modules or platforms, and adaptive modules. The definition of a number of different concepts allows for the identification of possible solutions, including defined interfaces and scenarios. An important aspect of this step is the integration of concepts to a fully functional product [Salhieh & Kamrani 1999]. Methods for synthesis are discussed in chapter 5.5, while methods for the identification of modules will be discussed in the following sections [Simpson et al. 2001].

[78] Other authors provide comparable procedures, which are referenced within the following discussion [compare for example Suh et al. 2007, Diaz 1998, Kusiak 2002, Simpson et al. 2001].

The overall *evaluation* allows for the choice of the best-suited solution, based on a defined business case and the application of different evaluation measures (compare chapter 5.6), identification of necessary compromises and an optimal solution. In many cases, it is the determination of costs that is referenced for evaluation, for example by De Weck & Suh, while other perspectives include time, reliability, quality and manufacturability, according to Kusiak [Suh et al. 2007, Kusiak 2002]. Additionally, metrics for the evaluation of modularity or change to define module boundaries can be applied to support the decision-making process [Martin & Ishii 2002].

The *implementation* as last step includes the organizational and technical design solution of the chosen approach, taking into account crucial change management steps and the expert knowledge of design departments.

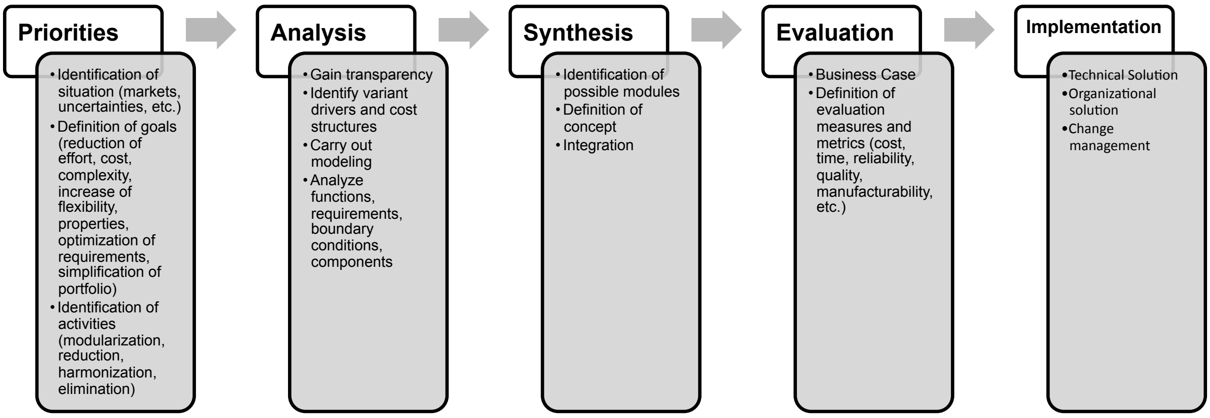

Priorities
• Identification of situation (markets, uncertainties, etc.)
• Definition of goals (reduction of effort, cost, complexity, increase of flexibility, properties, optimization of requirements, simplification of portfolio)
• Identification of activities (modularization, reduction, harmonization, elimination)
Analysis
• Gain transparency
• Identify variant drivers and cost structures
• Carry out modeling
• Analyze functions, requirements, boundary conditions, components
Synthesis
• Identification of possible modules
• Definition of concept
• Integration
Evaluation
• Business Case
• Definition of evaluation measures and metrics (cost, time, reliability, quality, manufacturability, etc.)
Implementation
•Technical Solution
•Organizational solution
•Change management

Figure 5-35 Generic process of variant management [compare Renner 2007, pp. 100-110]

The procedure as proposed includes the steps of prioritization and implementation, which are disregarded by other procedural models or methods [e.g. Farrell & Simpson 2008, Suh et al. 2007]. Since different methods are usually intended to suit a single purpose, the step of prioritization is still not considered. Implementation, on the other hand, is a major challenge to be considered in future projects [Simpson 2004], particularly in organizational or change management projects.

Models

In variant management, different models occur due to the different goals and methods. As was discussed in chapter 4.2, graphical, tabular, textual or analytical models come into question for modeling product architectures. In variant management, product architectures and product architecture families can be discussed by similar means. As a proposition was already made in chapter 4.3, the following sections will not discuss the possibilities of modeling in general, however, they will use models in graphical or matrix form as examples, according to the product architecture model. The discussed models are intended not to repeat discussions on product architecture modeling from previous chapters, but rather focus on product family specific entities and properties. In order to point out characteristics of product families in variant management, the following section cannot present a complete or exhaustive overview.[79] To give an impression, JIAO et al., for example, enlist numerous modeling approaches, ranging from UML, graph representations, set-based models, knowledge-based systems including rules and constraints, matrix-based and diagrammatic approaches, and computer-based and parametric models [JIAO et al. 2006].

As when considering product architectures in general, different views of the product family are feasible. JIAO & TSENG differentiate between the *functional*, *technical* (feasibility), and *physical* (manufacturability) views [JIAO & TSENG 1999]. *Functional decomposition* and *variant trees* are very common [see e.g. FOTSO et al. 2007], but have recently been replaced by more comprehensive and structural approaches, discussed in the following "methods"-section. Modeling of functional and physical product architecture was discussed in chapter 4.2 and is equally valid for the discussion of product architecture families. Different authors detail the view of manufacturing, focusing on the production view of variant management. DE LIT & DELCHAMBRE focus on product assembly and the impact on product family design. The goal of the approach is an appropriate design of product families, which systematically considers the assembly of the product variants. DE LIT & DELCHAMBRE define the product family as a group of products with large similarities of the *design concept*, the *main function* and the *assembly process*, within which a product variant is a (type of) product belonging to the family [DE LIT & DELCHAMBRE 2003, p. 108]. In their work, they point out a number of relevant entities of the product architecture, wherein the hierarchical interconnections represent "part of" relations. The core understanding and prerequisite for successful product family design is to perceive the product family and the underlying assembly process as that of one unique generic product, i.e. the whole product family has to be the focus of both the design and production processes [DE LIT & DELCHAMBRE 2003, p. 95]. Further principles include the late definition of single variants in production, constant iteration and recursion, as well as the quick and efficient exploration of the solution space [DE LIT & DELCHAMBRE 2003, p. 95]. For that purpose, DE LIT & DELCHAMBRE provide a hierarchy of the physical product structure divided into the *assembled product*, *subassemblies* and *components*, wherein subassemblies are what other authors perceive as modules, due to the strong interconnectedness of the subassembly's components. Subsets of the physical product architecture pose a set of components necessary for the product's integrity. The second pillar of the approach is the functional perspective of the product, resulting in the decomposition of the product's main function into technical functions, necessary for the product's internal integrity, and *functional subsets* and *functional subassemblies*. Functional subsets are components fulfilling at least one technical function and functional subassemblies are subassemblies fulfilling at least one technical function of the product family [DE LIT & DELCHAMBRE 2003, pp. 106-107].

[79] For more comprehensive overviews, see the following [BONGULIELMI 2003, FIXSON 2007, JIAO et al. 2006, RENNER 2007, SIMPSON et al. 2006]

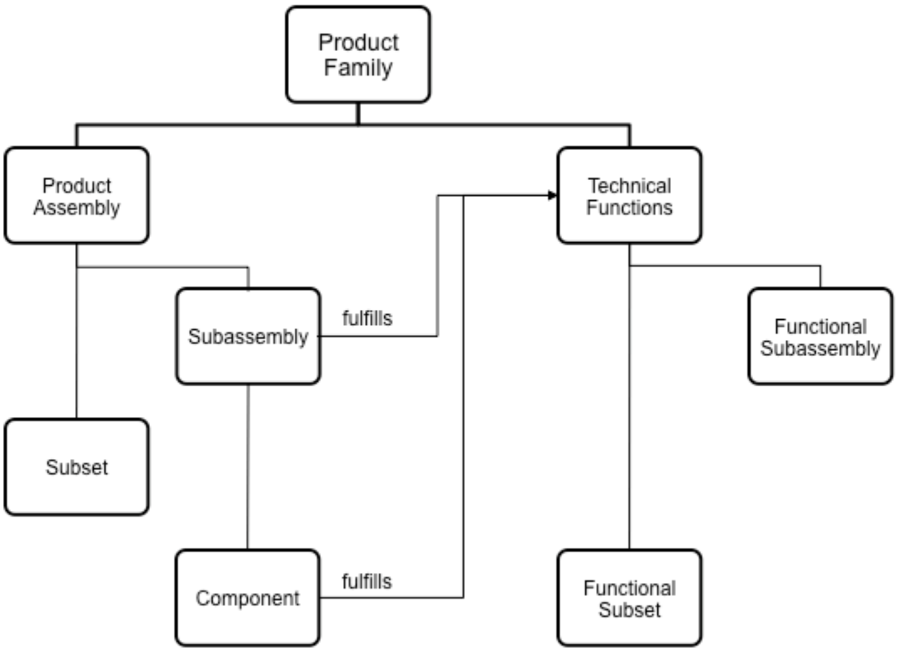


Figure 5-36 Structure of the product family from the perspective of assembly design [DE LIT & DELCHAMBRE 2003, pp. 106-107]

To complete the principles of a product family, DE LIT & DELCHAMBRE classify the entities of the product architecture as generic and variant entities from the physical and functional perspective. As a result, the concept of "*functional entities*" evolved, where common functional entities are part of all product variants (i.e. generic) and specific functional entities are part of specific product variants only [DE LIT & DELCHAMBRE 2003, pp. 109-117]. For the representation of the product family, DE LIT & DELCHAMBRE use graphs, in which the different classes are represented by the means of differently depicted nodes and edges [see e.g. DE LIT & DELCHAMBRE 2003, p. 127]. To cope with the product family and optimize the product portfolio, DE LIT & DELCHAMBRE introduce three independent indices for complexity, structure and standardization [DE LIT & DELCHAMBRE 2003, pp. 136ff., compare also SIDDIQUE & ROSEN 1999]. The discussed approach points to the relevant entities of the product architecture in the context of product families. The methods used, though in close relation to assembly, give hints about the potential and promising models, presented in the case graph structures and supported by the inherent mathematical possibilities. The choice of the model is thus supported, allowing for the integration of quantitative measures, such as indices, for the evaluation of different product architectures and designs.

Methods

Before the discussion of methods, the different general approaches often stressed in literature are worth mentioning. The different approaches, though inheriting different paradigms, are all based on the idea of modularity to achieve the discussed goals. The different approaches are summed up by RENNER as modular building blocks, common and repeat parts, commonalities, modularity in general, platforms, product series, or variant design [for a more extensive discussion see RENNER 2007, pp. 66 ff.]. Although the different approaches make use of the discussed leveraging strategies, a precise differentiation remains challenging. Depending on the chosen approach, priorities in terms of goals and leveraging strategies differ, yet cannot be discussed separately from one another [RENNER 2007, p. 79].

The following sections discuss methods for variant synthesis, analysis, metrics, and modularization, thus addressing the key aspects of the variant management process.

For **synthesis**, *graph grammars* provide the means for variant management as well, due to the similarity to the process of deriving possible products from a set of entities (see chapter 5.5.7). For the definition of variants, according to DU et al., the identified and predefined modules are coupled, due to the inherent rules. A characterization of modules, as well as their interfaces, enables the synthesis of variants. The possible combinations of modules (*configurational*), as well as the variation of element attributes (*scalable*), cause the product family variety. Required for successful application is the definition of a generic product structure, stretched across a number of abstraction-levels [DU et al. 2002]. Similar to the discussion of graph grammars for synthesis, the difficulty of application in the context of variant definition lies within the precise and elaborate characterization and definition of rules beforehand. Although the thorough analysis and definition is necessary, minor flaws cause incorrect outcomes, which are hard to compensate for and identify.

Different authors propose variant definition on the basis of *FBS systems.* JIAO & TSENG identify the functional, behavioral and structural view as core elements and respective entities of the product (family) architecture. The functional features allow for the analysis and definition of functionality (functional view), while the technical parameters represent the technological feasibility (behavioral view) and the components and/or assemblies lead to answering the questions of manufacturability (structural view). For an efficient application, the paradigms of modularity and commonality are highly relevant. Modularity represents the decoupling of architecture entities, while commonality characterizes the clustering of similar entities or modules into classes. Both concepts apply on all levels of the FBS-system, i.e. on functional, behavioral and structural views, resulting in e.g. functional and behavioral modules, as well as component or functional feature classes etc. [DU et al. 2001]. UMEDA et al. discuss the application of the Function-Behavior-Structure in the context of upgradeability and provide a method for the deletion and addition of FBS-fragments. The principles of configuration and scalability are stressed as well, since fragments are added or deleted due to the demand of new functions, or attributes of FBS-fragments are changed to suit moving targets in customer requirements which do not require the addition of a new function [UMEDA et al. 2005]. KUMAR & ALLADA add the customer needs to the FBS model in order to derive customer-oriented variants based on the given FBS-platform. To support that process, an ant colony optimization algorithm is applied to identify how the function and behavior structure should be composed. The approach shows similarities to QFD, interrelating customer needs, functions and behavior. The product architecture, in terms of the physical representation, is not considered in this approach. Limitations occur, in the sense that no new functions can be generated based on customer needs, and that the demand has thus to be known a priori. The approach of UMEDA et al., for example, aims to compensate exactly these shortcomings [KUMAR & ALLADA 2005, UMEDA et al. 2005].

Since the aspect of customer integration and coupling of **customer requirements** to the product architecture is a key factor in variant management, other authors stress this assumption as well. BONGULIELMI et al. discuss the K- & V-Matrix[80] in the context of variant management to depict configuration knowledge and thus support the sales processes. For that purpose, the authors chose the intersection of customer perspective and technical modularization as core aspects of the method [BONGULIELMI et al. 2002]. Based on a reduced range of domain mapping logics,[81] the application result displays the modular interdependencies of architectural choices from customer or technical view. Thereby, in contrast to variant trees, the K- & V-Matrix depicts the possibilities and validities of variants overlapping in one matrix, similar to the approach chosen by BRAUN & DEUBZER [BRAUN & DEUBZER 2007]. DEUBZER et al. use the domain mapping logics to interrelate physical or functional entities of the product architecture, not by their direct interrelations, but by the customer demand of ordering the features in combination. The result is depicted in the graph of functions, where the most ordered features are depicted in the center of the graph (i.e. a possible standard module or platform), while the relatively seldom ordered features (i.e. upgrade modules or specification modules) are aligned on the outside [DEUBZER et al. 2008].

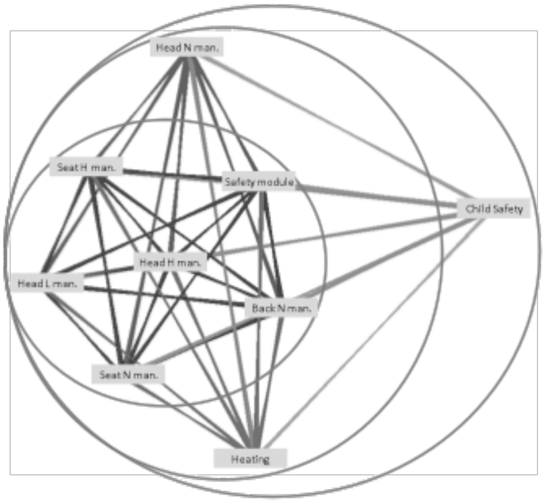


Figure 5-37 Coupling of functions of a car seat, based on customer order-behavior [DEUBZER et al. 2008]

For the **evaluation of concepts**, synthesized on the basis of customer needs, a number of measures and metrics can be applied. The different metrics are more or less applicable in different situations, depending on the goals of variant management, the product, and especially the type of cost calculation and available information. SIMPSON gives a brief overview of suitable metrics and available sources, as well as STRYKER & JACQUES or GERSHENSON et al. [GERSHENSON et al. 2004, SIMPSON 2004, STRYKER & JACQUES 2009]. Three main groups of metrics can be identified in variant management; namely the *commonality indices*, *structural metrics*, and *cost structures*. Since cost structures are largely

[80] BONGULIELMI provides a more elaborate discussion of the concept [BONGULIELMI 2003, pp. 57ff.].

[81] For an exhaustive discussion, see chapter 5.4.3.

dependent on the type of cost accounting and other boundary conditions, the following sections discuss examples of the first two groups.[82]

A comprehensive overview on commonality indices is provided by THEVENOT & SIMPSON [THEVENOT & SIMPSON 2006]. To give additional prospects, the Generational Variety Index will be introduced in the following section, focusing rather on the likelihood of change to identify robust elements of a product architecture. MARTIN & ISHII discussed the Generational Variety Index (GVI) to indicate the likelihood of changes of system elements over time. The definition begins with a modification of QFD, in which the likelihood of changes of customer demands is estimated, and then transferred to the system elements. The additional coupling indices (CI) for receiving (CI-R) and supplying (CI-S) relationships between components support the decision-making process by indicating whether or not a component is likely to change due to customer demands (GVI) or impact from other components (CI-R). The index for supplying relationships (CI-S), on the other hand, identifies components or modules, which can be customized according to customer demands, without causing impact on other system parts [MARTIN & ISHII 2002]. The identified indices for receiving and supplying relationships represent a similar perspective, as do the active and passive sum or activity and criticality as structural criteria, to analyze structural complexity [compare Maurer 2007, pp. 199-200 and 206]. Commonality indices, as discussed by THEVENOT & SIMPSON, mostly relate the common parts (takeover) and distinct component parts (unique), architecture structure levels, absolute number of components, unique and common interrelations between components, cost, volume, etc. [THEVENOT & SIMPSON 2006].

Graph Theory and network theory provide many metrics to characterize systems (see chapter 5.4.3). Since modularity is a structural issue, from the perspective of configuration rather than scale, a number of metrics allow for capturing the complexity and modularity (compare also the discussion of structural characteristics as evaluation criteria in chapter 5.6.3). Examples for the application of structural characteristics can be found in literature, for example by SOSA et al., who explicitly utilized a number of traits to characterize modularity of product components [SOSA et al. 2007]. In particular, SOSA et al. modify the metrics for degree (the number of connections an element owns), distance (the indirect dependencies an element owns), and bridge (the number of times an element appears within the path of a couple of other elements) and apply them to the product architecture. The respective indices incorporate the actual value for each node, divided by the maximum possible value for the architecture. As a result, the architecture entities can be ranked and compared, depending on the inherent "modularity", defined by the criteria degree, distance, and bridge [SOSA et al. 2007].

The **identification of modules** within the product architecture is usually conducted by structural analysis using DSM-methods, as introduced in chapter 5.4.3. Different authors discussed and applied the approach, making DSM-based clustering the standard approach for system analysis for identification of modules [see for example BROWNING 2001, KUSIAK 1999, p. 259ff, KUSIAK 2002, LINDEMANN et al. 2009, pp. 185ff., JIAO & TSENG 1999].

[82] GAHR provides an overview of cost accounting in the context of highly individualized products [GAHR 2006]; an overview on structural metrics is given in chapter 5.4.3.

Lifecycle perspective and stakeholders

The product lifecycle poses vital challenges for the management of product architectures (see e.g. [STARK 2005, pp. 55ff.]). Whereas the requirements of the use phase, in the form of variant management, were discussed in the previous chapter, the following paragraphs point out further requirements, stemming from different phases of the use cycle.

The lifecycle itself can be perceived as a sequence of work phases, each resulting in a progressed status of the product [HEPPERLE et al. 2009a]. The main phases of the product lifecycle, according to different authors, can be summed up as product planning, development and design, production (planning), distribution, utilization, maintenance, modernization, disposal and recycling [HEPPERLE et al. 2009a, compare ARNOLD et al. 2005, VDI 2221]. The different resulting states of the product show the product concretization during the phase of development and design. The states, which are usually referred to in literature, are similar to those resulting from models of the development process [see e.g. VDI 2221]. In detail, they are summed up as the product requirements, functions, working principles, components, prototypes and product documentation [HEPPERLE et al. 2009a].

From an entrepreneurial perspective, the lifecycle value of a product is the measurement of a product's success over a long time period. HONOUR & BROWNING set up a measurement system for the lifecycle value, resulting in a number of steps to proceed, including the identification of relevant stakeholders and key parameters [HONOUR & BROWNING 2007], which in turn are as relevant for the product architecture design as they are for the product value. Stakeholders mentioned are purchasers, users, activists, maintainers, owners of interfacing systems, firms operating the system, suppliers, providers of alternative systems, communities with certain interest in the product and firms operating the system infrastructure or providing complementary systems and services [HONOUR & BROWNING 2007]. The identification of the product stakeholders is just as crucial of a step in the context of this work as it is for the definition of a system's lifecycle value. To establish the link between the technical product and its requirements, HONOUR & BROWNING introduce key parameters, divided into the classes of benefits and sacrifices, wherein benefits are what the stakeholders gain, while sacrifices represent the resulting shortcomings for the stakeholders (compare also the differentiation between characteristics and properties as in [WEBER 2005a]). Examples of stakeholders mentioned are corporate management, employees, shareholders, subcontractors etc. [HONOUR & BROWNING 2007]. To complete the number of stakeholders, different possibilities exist, for example adopting the different stakeholders from the lifecycle perspective, yet it is as crucial undertaking to introduce the stakeholders early in the process and with appropriate methods, as it is to identify their needs, i.e. the expected benefits and sacrifices. The integration of different stakeholders, the appropriate methods for their incorporation and the crucial phases of integration during the lifecycle are discussed in numerous publications, which above all consider the integration of the customer as a stakeholder (for an overview, see e.g. KAIN et al. 2009).

In the context of product lifecycle management (PLM), product data management (PDM) is usually closely connected as enabler of the product lifecycle management itself [STARK 2005, pp. 233ff.]. Product data management represents the storage and information management of the product data, utilized and extended with further assets, such as the workflow management, coping with multiple versions of data, etc. [STARK 2005, p. 243].

The challenges in product lifecycle management can be summed up as the consideration of relevant phases of the lifecycle and resulting states of the product architecture, the integration of stakeholders and their needs (as discussed in chapter 5.3.1), the coping with changes, both reactive and proactive, and the management of the information relevant for all lifecycle phases.

5.7.2 Conclusion

The existing methods allow for the variant management, each from a certain perspective, to achieve modularization, optimize cost, reduce development expenses, eliminate variants etc. Projects with industry have shown that companies lack the ability to enable transparency for all stakeholders involved, as well as being deficient in gaining an overview over the complex interdependencies between the different aspects of variant management. The overall optimum solution is thus seldom acquired. The cause-and-effect chains containing domain-spanning linkages, in particular, are hard to grasp intuitively, and knowledge about those linkages and related effects barely exists. The dynamic of the whole variant management system shows in the impact of activities within one domain on other domains. For example, the management of the variants offered within the product portfolio from a sales perspective might allow for the satisfaction of the customer needs and thus require no further optimization. The technical system in terms of carry-over parts, basic modules, interfaces of modules, manufacturing etc. might nevertheless require optimization to allow for the desired company profit.

As for the management of product architectures, the lifecycle perspective and the stakeholders within it pose a large constituent for the process of decision-making during design. The challenge for the systems architect is to interrelate the product architecture with the demands of the stakeholders and lifecycle, in a way that the resulting product is introduced successfully and maintains its value over time. The translation of the needs of lifecycle and stakeholders into product properties, i.e. measurable requirements, is the main challenge to face. Additionally, the translation of needs and their fulfillment has to be monitored and controlled over time, thus supporting the iterative and recursive process of design.

5.8 Overall requirements to the solution

After the discussion of the existing approaches to the separate steps of product architecture management, the following sections point out overall requirements to the solution to be defined, which cannot be solved by the combination of existing approaches, methods and models. Chapter 5.9 will then combine all of the requirements, based on the discussion of the sections in chapter 5 and will conclude with the complete description of demands to the solution, for which chapter 6 provides the groundwork, while chapter 7 describes the overall approach to the management of product architectures.

The description of the situation in chapter 1 and the discussion of approaches to complexity and product architecture management in chapters 3 and 5 pointed out that the successful coping with product architectures requires a consistent support of the product lifecycle throughout all phases.

As such, **consistency** is required, not only along the process, which is to be supported, but also for the detailing of the product architecture entities along the process.

Given the consistency throughout the process and detailing of product architecture entities, the approach has to be able to capture the different discussed entities of product architecture, as well as the given goals in the context of different product architecture projects. This **comprehensiveness** is to be achieved by the consistent consideration of the process, as well as the capability to cope with the upcoming relevant entities and the purposeful interrelation of those. A solution is required that combines existing methods according to the respective goals and circumstances [compare WEBER 2005a].

As a last overall requirement of the solution, **flexibility** is required to allow for the capability to adapt to different project goals and available information, thus not forcing the user of the approach to gain all possible information to be able to apply the method. As a second cause for the requirement of flexibility, the adaptability during application to new situations has to be granted. The need for adaptation can arise from recursive or iterative procedures resulting in new results in different domains, or the change of product architecture entities, such as requirements or technologies along the application of the approach.

5.9 Conclusion

Based on the previous discussions and elaborations, the solution requirements can be summed up at this point, identifying the solution requirements and proposing means of addressing the remaining gaps. The overall requirements of the solution were discussed at the end of the previous section and in chapter 1.2, and are summed up as:

- **Consistency** (support of recursive and iterative procedures);
- **Comprehensiveness** (consideration of different relevant entities on different levels of concretization and incorporation of stakeholder perspectives);
- **Flexibility** (modeling approach to couple existing methods and models, based on an adaptable procedural model).

As a result, the approach is intended to enable:

- The capturing of **reactions between levels of abstractions and across domains** (against the background of different goals);
- The **disintegration of existing hierarchies** within domains;
- The search for **solutions on all levels**, both hierarchical (*consistency*) and on the basis of different entities (*comprehensiveness*).

6. Constituents of the solution approach

The remaining challenges for the management of product architectures require the addition of a number of constituents to enable the existing methods to interrelate, and to fulfill the requirements for the synthesis of product architectures. The following sections propose a number of additions to existing solutions, with respective examples, to bridge the remaining gaps. First, the modeling of existing approaches, previously discussed in the context of functional modeling, is presented in combination with the MDM approach, enabling the transfer of existing methods into a generic notation. Second, the coping with hierarchies and underlying paradigms is discussed, proposing a procedure to incorporate hierarchical considerations into an overall approach. To support the synthesis on the basis of existing solutions across the entities of the product architecture, an approach is introduced to systematically establish a model for the synthesis of product architectures. Finally, the comprehensive modeling of the solution space is proposed, enabling a more comprehensive overview of solutions than existing synthesis approaches.

6.1 Modeling in MDM notation

The coupling of existing methods requires the transfer of the generic modeling information into MDM notation, to enable the application of the powerful means of analysis based on the MDM approach and Graph Theory. The procedure can be broken down into the following steps:

- Identification of entities (elements) of the method
- Identification of relations
- Decoupling of classes of entities and relations into discrete matrices
- Extension of information (if useful) using domain mapping

The first step includes the classification of the elements of the method. For the flow-oriented functional model, for example, the entities' states and operations exist. In the sense of domain mapping, it is important to separate the different classes from one another and identify the boundaries of each class precisely.

The same applies for the relations of model entities. For the case of the flow-oriented functional model, directed dependencies exist between operations and states. According to the semantics of the model, no direct relations exist between different states or different operations.

Once the semantics of the model are clearly defined, the content of the model can be transferred into the MDM model, incorporating all of the information from the model. For the given example, the DMMs of states to operations and operations to states exist in the model, as is indicated as step 1 in the following figure.

As a last step, the DSMs of the model can be computed, indicated in steps 2 and 3 in the following figure. As a result, for the given example the DSMs of states and operations can be analyzed using the metrics and techniques described in chapter 5.5.6.

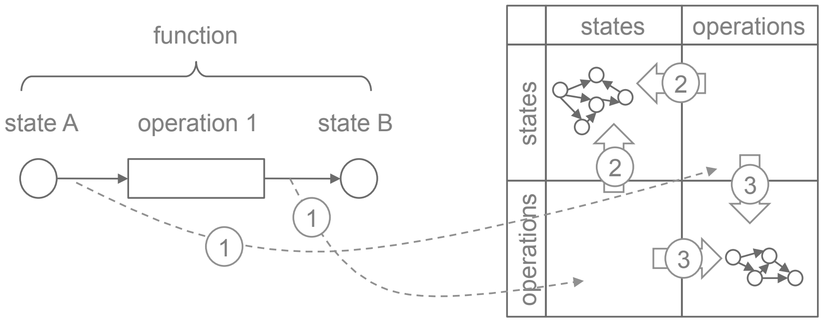


Figure 6-1 Transfer of a functional model into MDM notation

Based on these rather simple steps, many models of working methods, be they functional models as discussed, FBS systems, or the approaches of QFD or FMEA (compare chapter 5.6.3) can be transformed into the generic modeling approach of the multiple domain matrix. The usage of the modeling has to be discussed in each case, yet the transformation allows for the coupling of methods, as the following sections will show, and thus supports the continuity of information along iterative and recursive processes. As a second benefit, the models are accessible to the numerous mathematical and structural optimization and analysis approaches.

6.2 Coupling of methods and models

If the models of a method are transferred into matrix notation, the resulting matrix model can be interrelated with further existing models, such as relational functional models or components [DEUBZER & LINDEMANN 2008]. Outcomes of methods reflect directly on one another. In the case of the discussed functional models, for example, the operations pose the linkage between both functional models. The additional adding of components results in an even more complete picture, again enabling the coupling to methods such as QFD or FMEA, for example. The following figure depicts the MDM as a combination of relational and flow-oriented functional model.

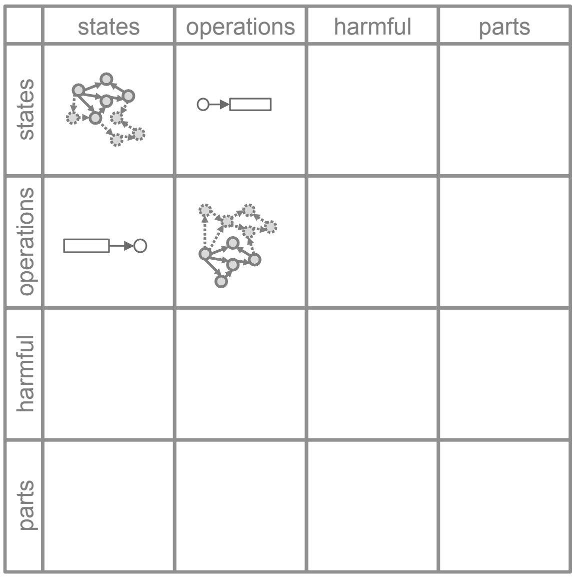


Figure 6-2 Coupling of different models in MDM notation (schematic)

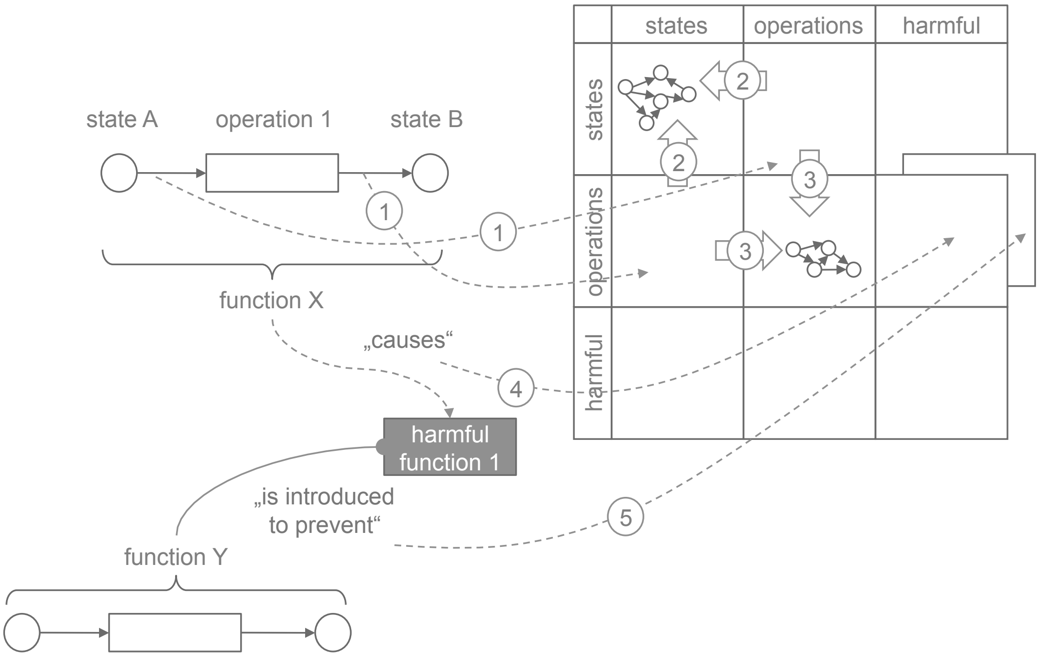


Figure 6-3 Coupling of different functional models in MDM notation (example)

6.3 Coping with hierarchies and recursive procedures

To efficiently cope with hierarchies and recursive procedures, hierarchies are required to be incorporated into the model. The dilemma of hierarchical visualization is the one-dimensional depiction of content. The following figure, for example, shows the hierarchical "part of" decomposition of the power-train functions of an automobile. Additionally, a few linkages from a networked perspective are added, pointing clearly to the limitations of the model. As such, only one domain is depicted here, yet the depiction of a networked view is already difficult.

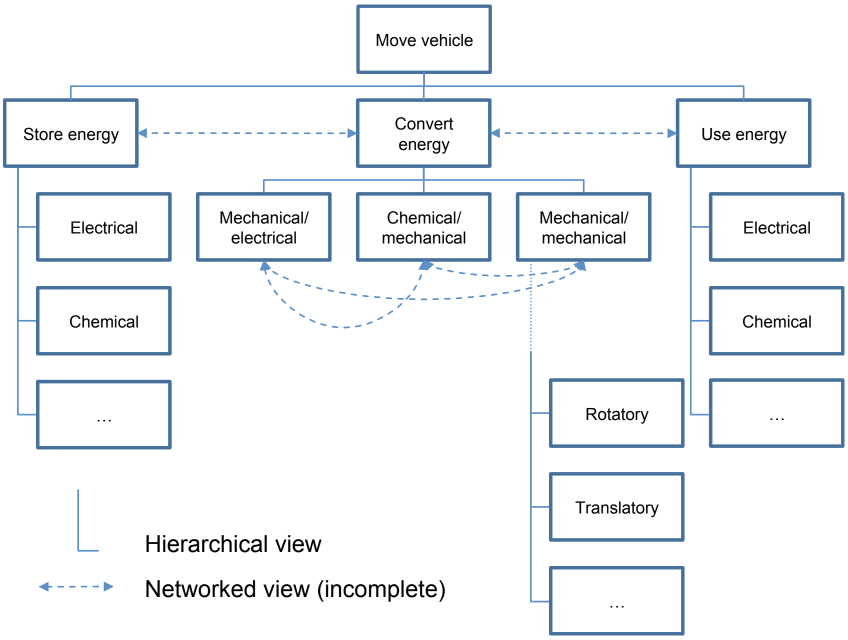


Figure 6-4 Hierarchical functional decomposition with denoted networked view

The DSM approach evolved over the years, incorporating different domains horizontally. Already a powerful means of systems engineering, DMMs were added to widen the scope of the approach. The integration and extension of both approaches within the MDM allowed for a comprehensive combination of domains and the active coping with them.

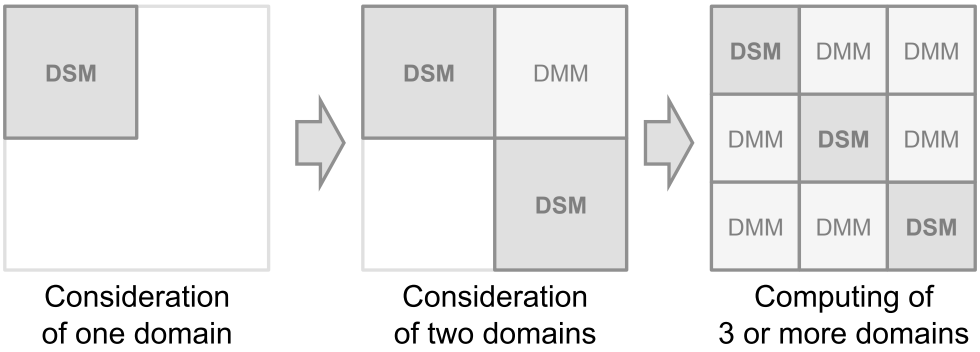

DSM
Consideration of one domain
DSM
DMM
DSM
Consideration of two domains
DSM
DMM
DMM
DMM
DSM
DMM
DMM
DMM
DSM
Computing of 3 or more domains

Figure 6-5 Development of matrix-based approaches

However, the application of the approaches requires a defined level of abstraction before information acquisition, which poses problems for the user. Information cannot always be consistently captured on a defined level of abstraction in all relevant domains, thus resulting in the need for compromises within the data quality. Since hierarchical views contain useful information (similar to categories or classes) that is lost in networked views and vice versa, a solution to incorporate both is necessary.

The goal of the presented approach is thus to incorporate both views into the model, and enable not only the application across domains, but also across levels of a domain's hierarchy. As such, the methods are able to provide application support throughout the processing of structures (e.g. the concretization of product architectures), and can overcome difficulties when dealing with hierarchies and networks of systems in parallel.

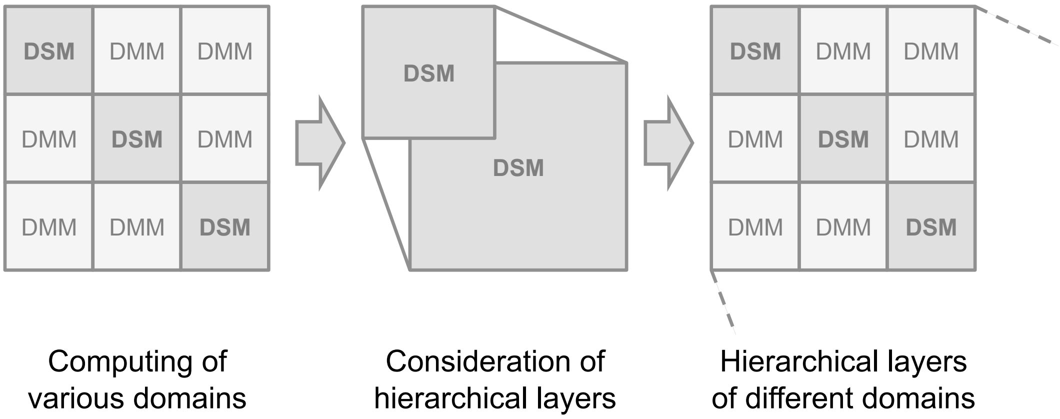

DSM
DMM
DMM
DMM
DSM
DMM
DMM
DMM
DSM
Computing of various domains
DSM
DSM
Consideration of hierarchical layers
DSM
DMM
DMM
DMM
DSM
DMM
DMM
DMM
DSM
Hierarchical layers of different domains

Figure 6-6 Adding the dimension of hierarchical layers to the MDM approach

The example of functional modeling is again stressed to explain the procedure. The following figure depicts the functional model of a drivetrain in a flow-oriented manner on very abstract level, and concretized on a more detailed level, in the sense that different energy types can be stored, converted and used.

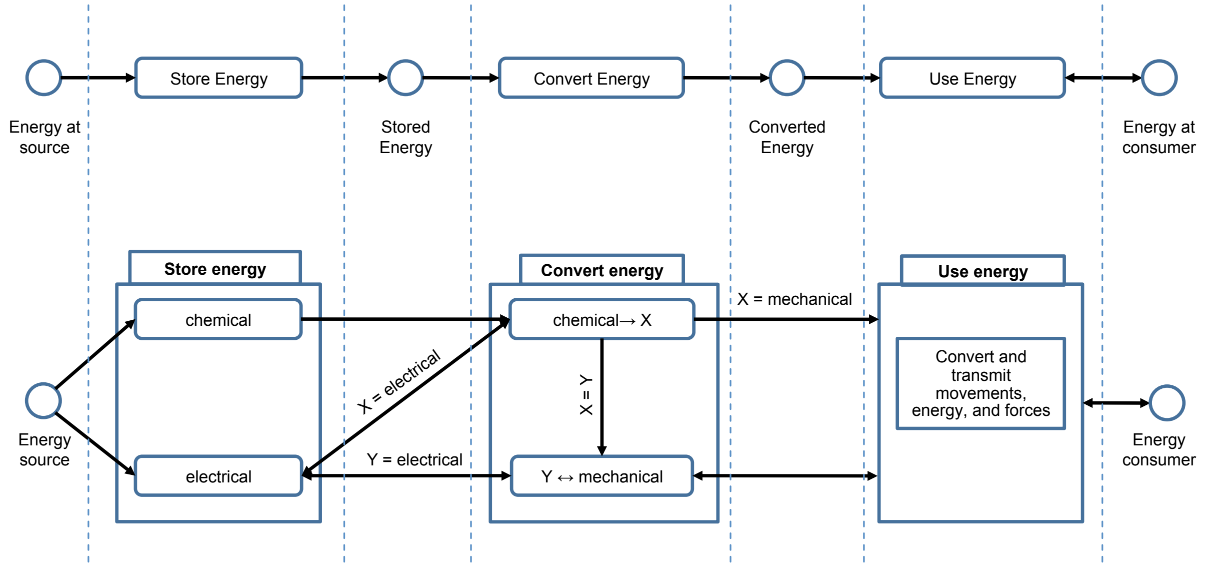


Figure 6-7 High level functional models

The depiction of an example of an automotive drivetrain is shown below, using concrete operations and states, which in the above models were generalized and combined to depict the higher-level view. Thus, the same system is shown in the different figures, yet on different levels of abstraction.

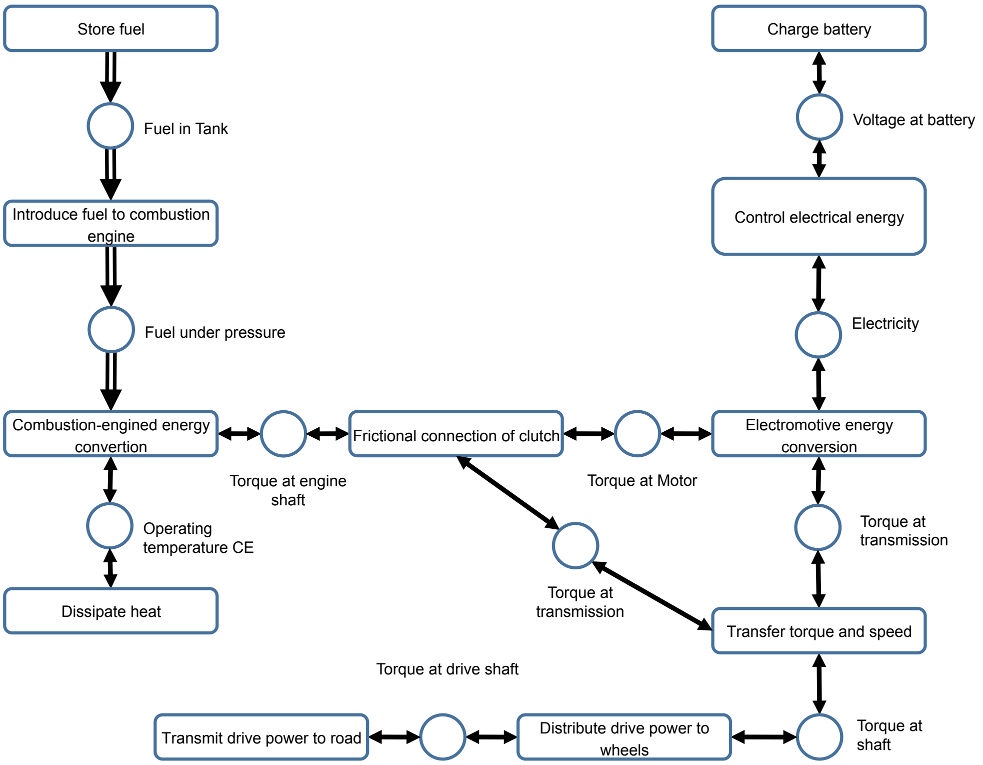

Store fuel
Fuel in Tank
Introduce fuel to combustion engine
Fuel under pressure
Combustion-engined energy convertion
Operating temperature CE
Dissipate heat
Torque at engine shaft
Frictional connection of clutch
Torque at Motor
Torque at transmission
Charge battery
Voltage at battery
Control electrical energy
Electricity
Electromotive energy conversion
Torque at transmission
Transfer torque and speed
Torque at drive shaft
Transmit drive power to road
Distribute drive power to wheels
Torque at shaft

Figure 6-8 Concrete functional model

The models depicted in the above figures all show information acquisition based on existing approaches, i.e. the flow-oriented functional model, differentiated by the levels of abstraction. It becomes clear that information in not available in all models, though all pose valid contributions for structural variation for design synthesis, for example. Recurring difficulties appear when integrating models (and their inherent information) for a comprehensive view.

Across the different models, a hierarchical model can be established, joining all three models in a four-level hierarchical structure. The hierarchy within is composed in the sense of "part of" relations. "Store energy", for example, is part of the overall function "move vehicle", while on the other levels of the hierarchy, chemical or electrical energy can be stored. In the given example, stored chemical energy is the storage of fuel, while the stored electrical energy is represented by the charging of the battery.

The hierarchical model provides a valuable overview of the system, yet neglects structural or networked information, and leaves numerous possibilities to couple the functions on level 4 for a fully functional drivetrain. As the introductory example showed, a combination of both views within this visualization turns out to be possible.

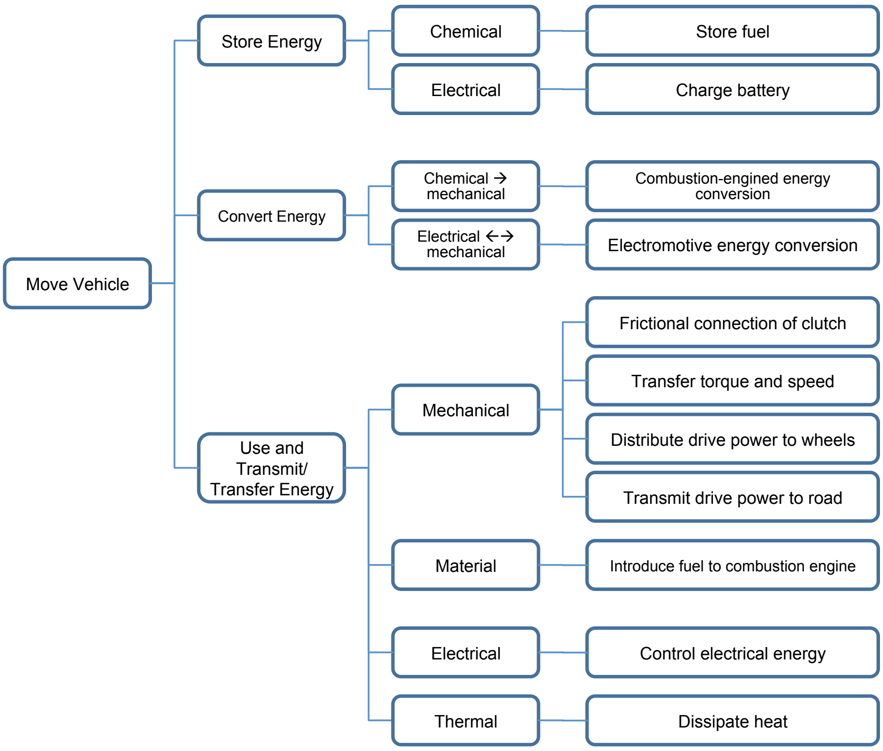

Move Vehicle
Store Energy
Chemical
Store fuel
Electrical
Charge battery
Convert Energy
Chemical → mechanical
Combustion-engined energy conversion
Electrical ↔ mechanical
Electromotive energy conversion
Use and Transmit/ Transfer Energy
Mechanical
Frictional connection of clutch
Transfer torque and speed
Distribute drive power to wheels
Transmit drive power to road
Material
Introduce fuel to combustion engine
Electrical
Control electrical energy
Thermal
Dissipate heat

Figure 6-9 Hierarchical functional model

The following steps propose a procedure to couple a hierarchical and networked view into the model. The core idea is the definition of hierarchical levels as distinct domains, and as such the modification of the MDM approach. The following figure depicts that process, showing the first benefit, namely the avoidance of parallel depiction of energy types (chemical, electrical etc.).

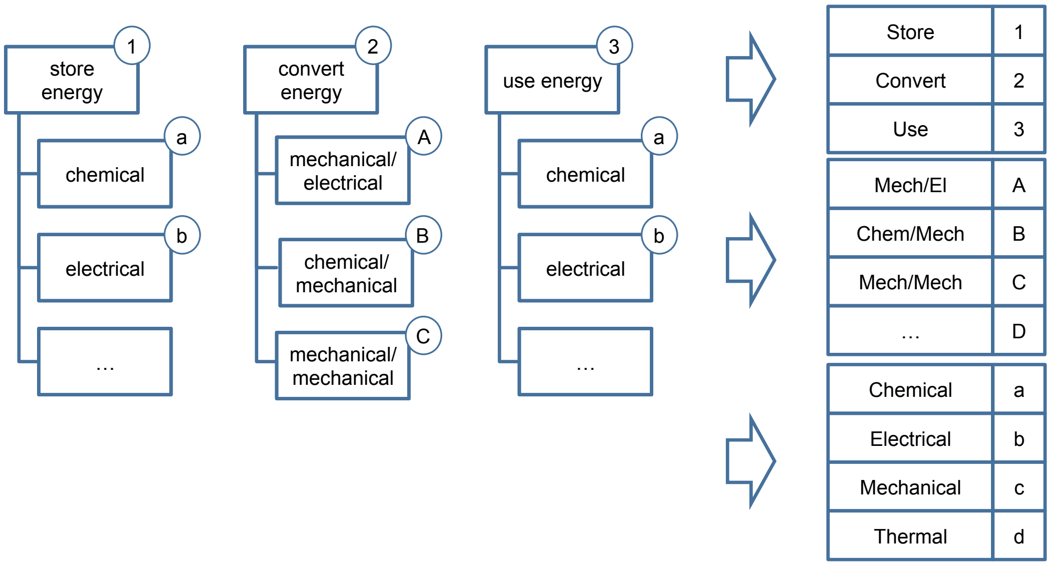
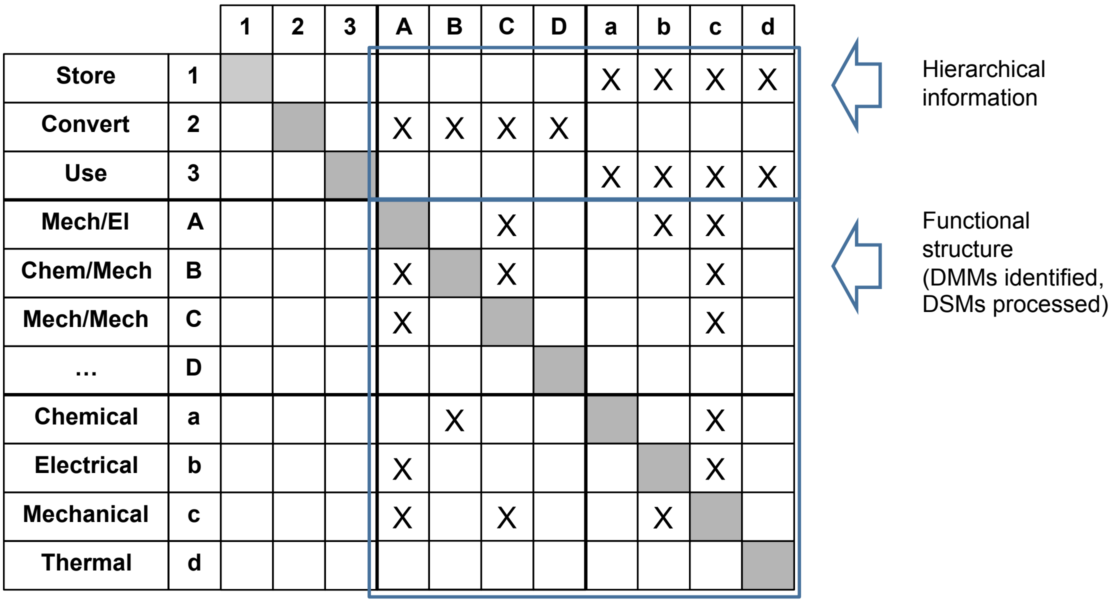


Figure 6-10 Establishing MDM domains

Based on the establishment of the MDM, the given information within the models can be incorporated as well. The following figure shows the level 2 and 3 hierarchies into three domains, allowing for the identification of relations within hierarchical DMMs, with the possibility of computing the DSMs within each domain.

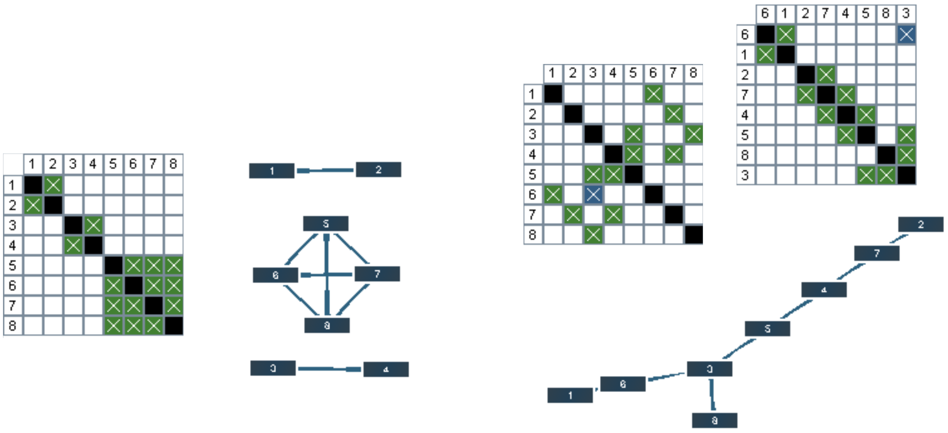


		1	2	3	A	B	C	D	a	b	c	d
Store	**1**	▨							X	X	X	X
Convert	**2**		▨		X	X	X	X				
Use	**3**			▨					X	X	X	X
Mech/El	**A**				▨		X			X	X	
Chem/Mech	**B**				X	▨	X				X	
Mech/Mech	**C**				X		▨				X	
…	**D**							▨				
Chemical	**a**					X			▨		X	
Electrical	**b**				X					▨	X	
Mechanical	**c**				X		X			X	▨	
Thermal	**d**											▨

Hierarchical information
Functional structure (DMMs identified, DSMs processed)

Figure 6-11 Completing MDM information

After the depiction of all inherent information, the computation of the domain mapping can be conducted across the different levels of the hierarchy as well, allowing for the identification of clusters in the sense of branches of the hierarchy, and, more usefully, the transfer of detailed models into higher levels of the hierarchy.



Functional level 3
hierarchical view
(View was computed using
hierarchical information given in
the level 2 functional domain)

Functional level 3
networked view
(View was computed using
networked information given in
the level 4 functional domain)

Figure 6-12 Computing hierarchical information and networked information on the functional level 3

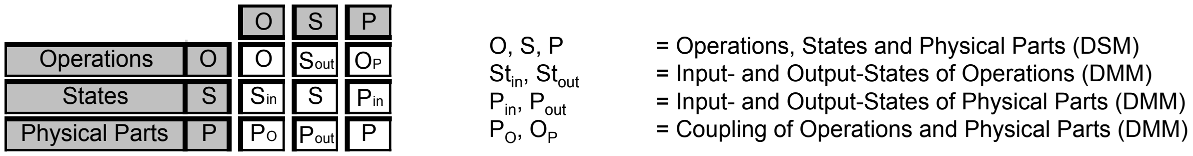
As the above figure illustrates, the depiction of hierarchies allows for the incorporation of different paradigms into the hierarchical model. These can coexist in an interrelated manner, depending on the chosen domain mapping (compare chapter 5.4.3). The left part of the figure shows the "part of" dependencies between functions, and the right shows a computed networked view within the same level of abstraction. Since the details stem from one concrete solution, the computed structure on level 3 depicts exactly that solution, not the complete possible linkages depicted in the original model. As such, the hierarchical information can serve as a filter between the different levels of abstraction, each of which contains different original networked data.

Of course, the approach is feasible in different domains (components, functions etc.), and can further help in uncovering inconsistencies in the model through comparison of hierarchical and networked views (e.g. by clustering). The integration of both views, hierarchical and networked, enables capturing the benefits from both perspectives. The calculation of matrices allows for a completion of yet-incomplete models on different levels, adding the networked information to formerly unconnected levels. The detailing of a model along the process is possible, as information can be computed across hierarchical levels. The following sections will make use of that capability. As a final benefit, the findings in later phases of the concretization process can be aggregated to the previous levels, allowing for navigation through the levels of abstraction.

6.4 Supporting synthesis of product architectures

To apply the framework for product architectures to synthesis, the functional modeling approach is stressed for the following example as well, following the trains of thought of renowned synthesis approaches (compare chapter 5.5.2). To allow for a consistent application of different methods, the MDM is used as a backbone for application in this case as well. Existing models can be transferred into matrix notation to allow for a consistent strategy across the levels of abstraction, as discussed in the previous chapters. The following sections will introduce the meta-model of the presented example and give an overview of the potential support of product architecture synthesis, based on the meta-model.

The meta-model of the MDM, depicted in the following figure, is chosen to clarify the support to product architecture synthesis, displaying the interplay of functions (as a combination of operations and states, as discussed in the flow-oriented functional model (compare chapter 5.5.2)) and physical parts.[83] Considering the nearly solution-neutral functional model as a starting point, the decomposition of a flow-oriented model is conducted, according to chapter 6.1, resulting in the operations and states matrices of the meta-model. The entities of the flow-oriented functional model are depicted in the domains “Operations” (O) and “States” (S).



		O	S	P
Operations	O	O	S_{out}	O_P
States	S	S_{in}	S	P_{in}
Physical Parts	P	P_O	P_{out}	P

O, S, P	= Operations, States and Physical Parts (DSM)
St_{in}, St_{out}	= Input- and Output-States of Operations (DMM)
P_{in}, P_{out}	= Input- and Output-States of Physical Parts (DMM)
P_O, O_P	= Coupling of Operations and Physical Parts (DMM)

Figure 6-13 Meta-model of the example

Accordingly to the approach of flow-oriented functional modeling, the definition of the required system functions can be conducted by identifying the necessary types of flow of the system (e.g. signal-, material-, information- or energy-flows). As varied types of dependencies can be distinguished (compare chapter 5.4.3), the different types of flow can be identified separately and integrated in a ΣMDM. The chosen level of abstraction in the given example is that of core technical functions concerning energy flows, while other types of flow were not considered in the presented case.

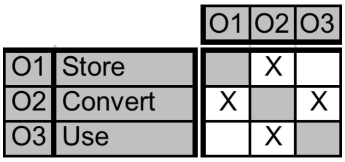
Based on such a functional model of the system, the synthesis can be conducted. In the present case, the use of existing physical parts (P) is chosen as an example. Existing physical entities (such as in-house solutions of predecessors, solutions of competitors etc.) can be assigned to the combination of operations and states, depending on the main operation and input and output states of the physical entities. Given a functional decomposition of the core system functions, the assignment of physical parts (solutions) can be considered similar to the composition of a morphological matrix (compare chapter 5.5.5).

[83] The integration of effects and working principles is discussed in the following chapter 6.5, completing the conventional synthesis approach.

The resulting DSM of physical parts (P) then represents – in contrast to other approaches – the solution space of technological solutions. Whereas other approaches allow for the analysis and definition of discrete solutions as a combination of a selected number of parts, the presented approach enables the depiction of the cross-linked components within the network of solutions, allowing for the identification of potentials, overlaps and restrictions within the solution space.

To clarify the introduced approach and meta-model, the following sections introduce an example of the application. Similar to previous examples, the drivetrain of an automotive vehicle was chosen, with the focus on the energy flow and the provision of relevant user functions.

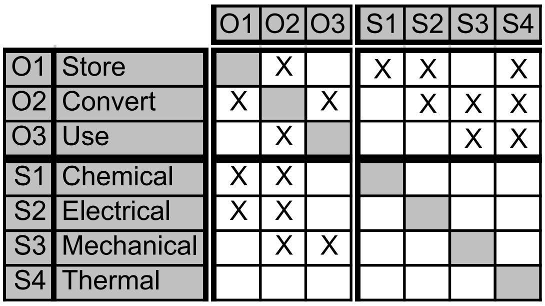
Identified user functions are “drive conventional”, “drive electric”, “boost”, and “recuperate”. From these, the core functions need to be derived on an abstract level. To describe a system on that level, a small set of rudimentary functions is sufficient to describe the system. The resulting operations “store energy”, “convert energy”, and “use energy” were chosen for the given example (compare the level 2 functional decomposition of the example in chapter 6.3). The marks in the matrix represent energy flows from row to column.



		O1	O2	O3
O1	Store		X	
O2	Convert	X		X
O3	Use		X	

Figure 6-14 Resulting operations (O) DSM

These operations can be detailed by the systematic combination with their possible input and output. For the chosen notation, two DMMs serve as an extension of the operations-DSM to depict their input and output in terms of energy types, with energy flows depicted from row to column.



		O1	O2	O3	S1	S2	S3	S4
O1	Store		X		X	X		X
O2	Convert	X		X		X	X	X
O3	Use		X				X	X
S1	Chemical	X	X					
S2	Electrical	X	X					
S3	Mechanical		X	X				
S4	Thermal							

Figure 6-15 Input and output DMM between operations (O) and energy states (S)

According to the arithmetic matrix operation given in the following equation 6-1, the possible conversions between different energy types can be depicted in a single DSM.

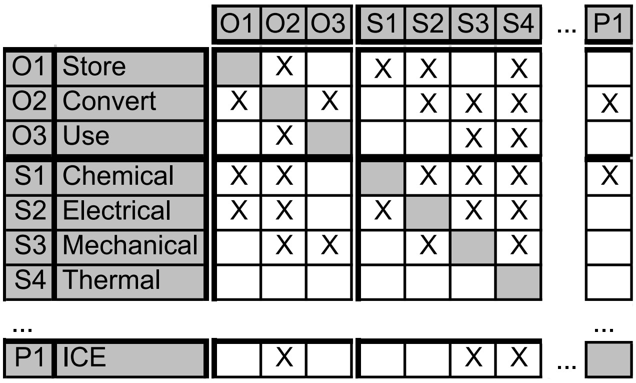
Equation 6-1 $$S = S_{in} \cdot O \cdot S_{out}$$

Based on the chosen input and output energy types, individual possibilities can be identified (for example, nuclear energy or radiation is not an issue, considering the system border in automotive engineering) to allow only valid solutions in the following steps. As depicted in the following figure including the resulting DSM of energy states, thermal energy output is not reused in the current systems considering the main energy flow. Of course, secondary systems, such as the cooling, are required to handle the thermal energy output. Considering the main energy flow, thermal energy might be reused for gaining electrical energy in future concepts. Structural characteristics as such can point out future improvements and priorities in innovative product architecture design and support the creativity during the design process.

		O1	O2	O3	S1	S2	S3	S4
O1	Store		X		X	X		X
O2	Convert	X		X		X	X	X
O3	Use		X				X	X
S1	Chemical	X	X			X	X	X
S2	Electrical	X	X		X		X	X
S3	Mechanical		X	X		X		X
S4	Thermal							

Figure 6-16 Derived DSM of energy states (S) low right

Technical subsystems are identified and coupled with operations and their respective input and output concerning the given energy states in the established MDM. The identification of technical subsystems can be conducted on the basis of predecessors and products of other product lines within the company, the benchmarking of competitors' products, and the screening of upcoming technologies. As an example, a conventional internal combustion engine is integrated into the model. The engine is hereby represented as a system element with the core function of converting energy, requiring chemically-bound energy as an input and providing thermal and mechanical energy outputs.



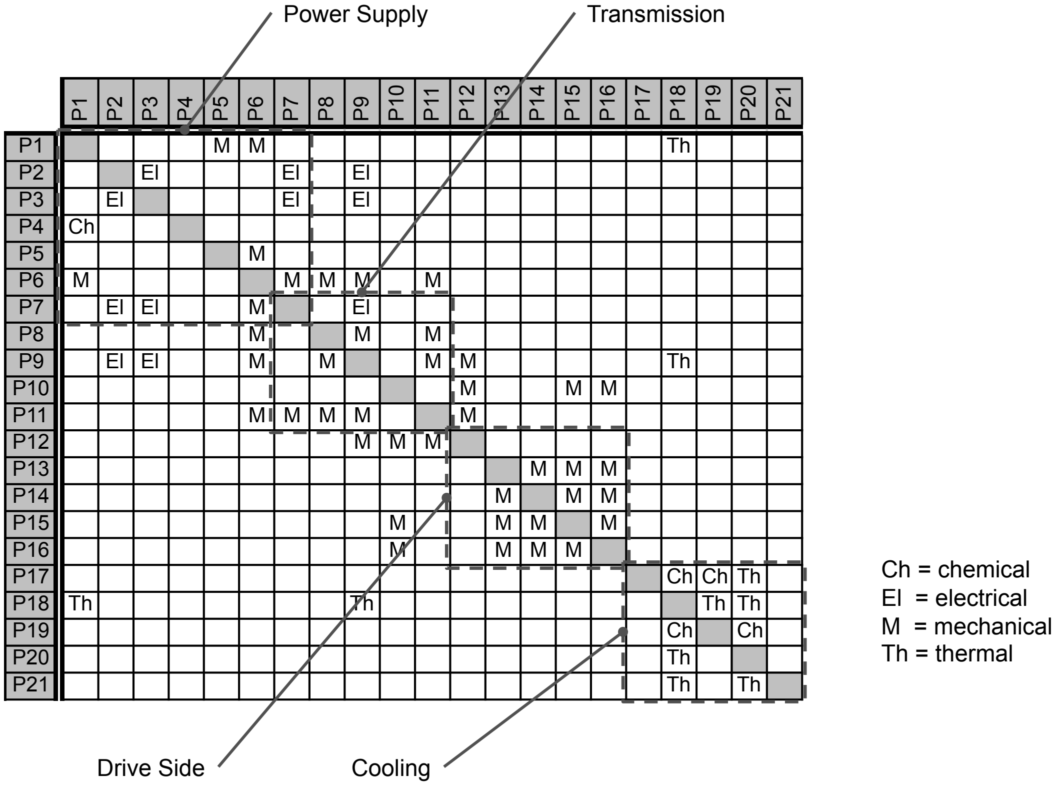
		O1	O2	O3	S1	S2	S3	S4	...	P1
O1	Store	▨	X		X	X		X		
O2	Convert	X	▨	X		X	X	X		X
O3	Use		X	▨			X	X		
S1	Chemical	X	X		▨	X	X	X		X
S2	Electrical	X	X		X	▨	X	X		
S3	Mechanical		X	X		X	▨	X		
S4	Thermal							▨		
...										...
P1	ICE		X				X	X	...	▨

Figure 6-17 Integration of Physical Parts (P)

Based on the given information, a matrix of physical components can be derived according to equation 6-2, showing the network of components concerning their energetic interfaces. As a result, the DSM of physical parts (P) shows the sum of all available physical parts and their combination.

$$\textit{Equation 6-2} \qquad P = P_{in} \cdot S \cdot P_{out}$$

The resulting DSM (P) of physical parts is depicted below. It shows the physical parts of three different existing drivetrain solutions, which were considered from functional and physical parts perspectives. The dependencies were marked within, indicating the energy type “chemical” (Ch), “electrical” (El), “mechanical” (M) or “thermal” (Th).



	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21
P1	▨				M	M												Th			
P2		▨	El				El		El												
P3		El	▨				El		El												
P4	Ch			▨																	
P5					▨	M															
P6	M					▨	M	M	M		M										
P7		El	El			M	▨		El												
P8						M		▨	M		M										
P9		El	El			M		M	▨		M	M						Th			
P10										▨		M			M	M					
P11						M	M	M	M		▨	M									
P12									M	M	M	▨									
P13													▨	M	M	M					
P14													M	▨	M	M					
P15										M			M	M	▨	M					
P16										M			M	M	M	▨					
P17																	▨	Ch	Ch	Th	
P18	Th								Th									▨	Th	Th	
P19																		Ch	▨	Ch	
P20																		Th		▨	
P21																		Th		Th	▨

Figure 6-18 Portion of the resulting DSM of physical parts (P)

Based on the information given, rules can be derived for the automated synthesis of valid solutions. For example, starting from one physical part with the core operation "use" to one physical part with the core operation "store", following necessary paths through the graph, which represents the numerous physical parts and their energetic interrelations. As a result, solutions are derived, which can be evaluated by their fulfillment of user functions represented by paths within the physical parts DSM. To give an example, the user function "drive conventional" requires a storage component of the type "chemical" with a link to a converter component of the type "chemical-mechanical" which again links to a related component to use this energy.

The above example has methodically shown possibilities for conducting the synthesis of product architectures, based on the methods introduced in chapters 6.1, 6.2, and 6.3. As was discussed, the major advantages of the approach are the systematic exploration of the solution space, on the one hand, and the continuous application of models and methods, allowing for the depiction of functions, physical parts, and their interconnectivity. The following chapter will discuss these exact advantages and point out further ways to interact and cope with the networked depiction of the solution space.

6.5 Depiction of the solution space

Based on the approach discussed in the previous chapter, the systematic synthesis of product architectures can be supported by systematically defining the logical dependencies of the solution space (in the given example functions and physical parts) for the generation of valid solutions, manual or automated. Different possibilities exist for the evaluation of solutions, based on the graph representation of the solution space. In the example presented, those evaluation criteria are user functions, degree of efficiency concerning the involved energy conversions and remaining potential in terms of “leaf nodes”, such as the mentioned thermal energy output. Additionally, a clustering of the solution space allows for a preliminary definition of desirable modules within the space of possible solutions.

The following figure shows the portion of the solution space, depicted as a matrix in chapter 6.4. Significant modules and characteristics of the graph appear, supporting the definition of subsystems to be discussed in the early phases of design. Since the solution space is depicted as a whole, clustering for variants can influence decisions for or against conceptual solutions at that early stage.

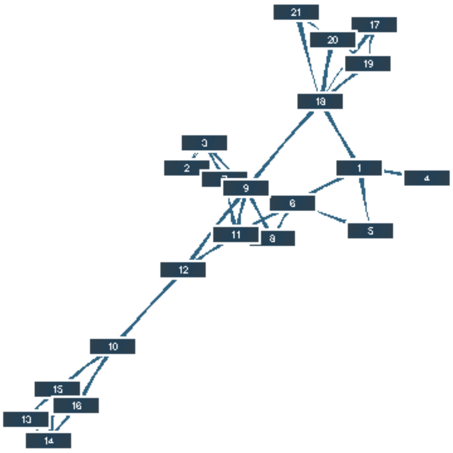


Figure 6-19 Portion of the solution space (P) in graph depiction

Constraints within the solution space can be depicted by the identification of structural characteristics within the solution space. For example, bridge edges (the encircled edge in the following figure) connect different modules of the solution space. If the connection is necessary in terms of the realization of the system’s user functions, the existence of the entities (in the given example physical parts) providing that edge is inevitable. If the side effects or properties of the given physical parts are unsatisfactory, a focused search for alternative solutions may be conducted. Alternative solutions should then allow for the substitution of undesirable components, and at the same time enable the required functionality.

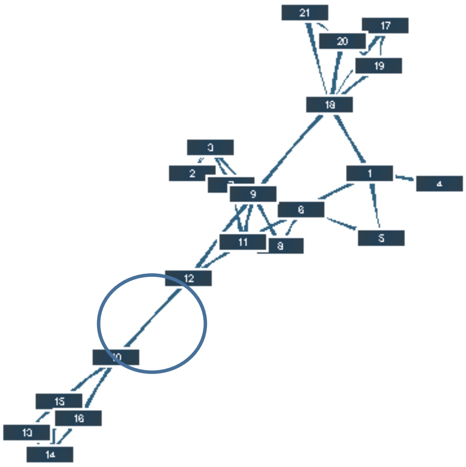


Figure 6-20 Identification of Constraints (P)

As well as constraints, potentials within the solution space can be depicted by the use of structural characteristics. In the given example, user functions can be translated to paths in the graph model of the solutions space. The user function "drive electric", for example, requires a complete path between relevant drive units, such as an electric motor, and the vehicles wheels (encircled in the following figure). The different paths between these nodes describe possible solutions and the node properties (such as degree of efficiency, cost, weight, part of a product family etc.), which characterize the solutions and support the process of decision-making and evaluation of solutions. The different paths can be identified based on generic graph-theoretical algorithms. By adding further technologies and components, the solution space widens and allows for the identification of potentials within. The number of paths increases, as does the number of known solutions and their improvements.

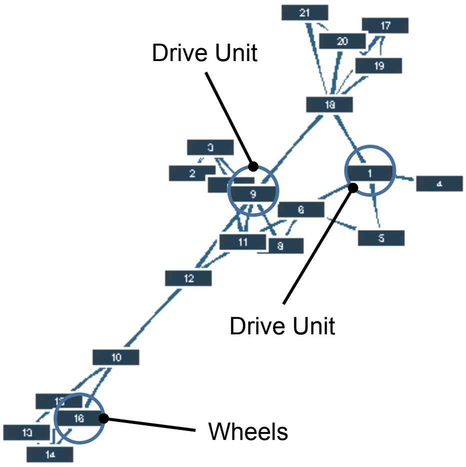


Figure 6-21 Identification of Potentials (Ph)

The identification and evaluation of valid solutions is conducted by establishing the present fulfillment of user functions, their requirements and quality. The recuperation of energy, for example, requires the existence of a path from wheels to battery, whereas the user function “boost” asks for the valid paths of “drive conventional” and “drive electric”. As in the identification of potentials, the different paths for required user functions can be evaluated, enabling the selection of most suitable solutions. In the given example, the choice of solutions and the identification of user functions are realized by the identification of energy flows within the product architecture. Other use cases might require focusing on other flows or depending on different characteristics of the product architecture.

So far, existing solutions of the product portfolio can be depicted within the solution space. Alternatives for physical parts or paths of energy flow within the solution space can be allocated. An alternative search for solutions can be conducted on the functional level through variations at the functional level, instead of or in addition to physical parts.

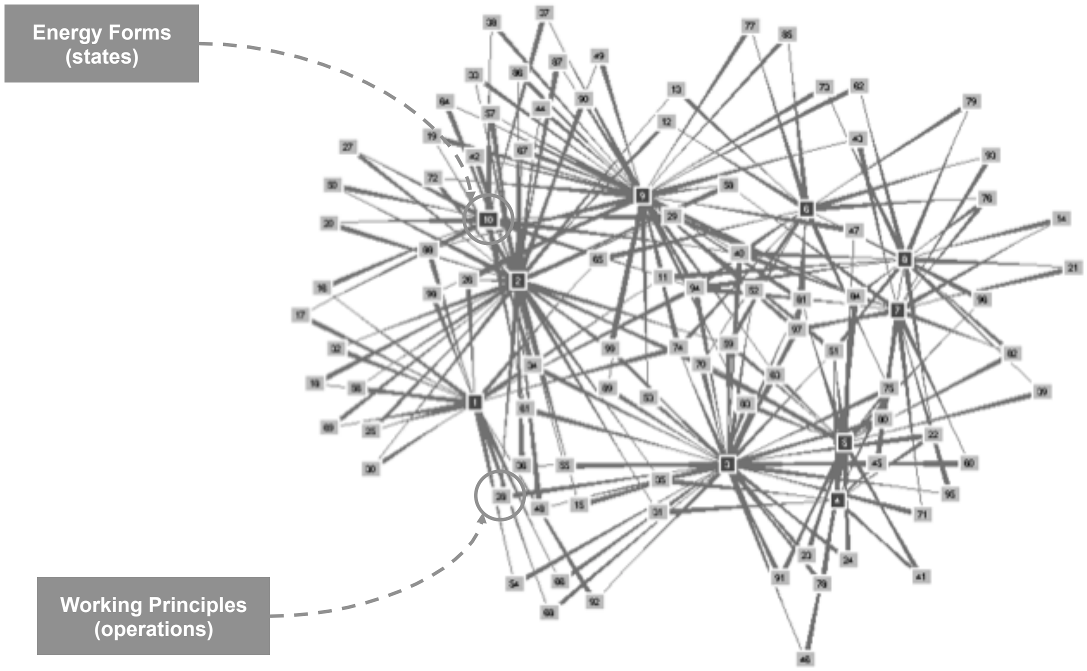


Figure 6-22 Depiction of solution space in graph form (from Deubzer & Lindemann 2008)

To exceed the level of synthesis in terms of the combination of known physical parts, the following sections aim for the support of the solution finding, based on the design of solutions, rather than the composition of architectural variants.[84] While radical innovations largely stem from the systematic variation of product architectures [compare Henderson & Clark 1990, p. 12], synthesis is commonly supported by functional abstraction and the following application of working principles to physically enable the realization of functions. The above figure shows a network of working principles (according to Lauer et al. 2008, compare also Graebsch et al. 2009 and Deubzer & Lindemann 2008), interlinked by their respective input- and output-forms, i.e. pneumatic or hydraulic pressure, electric charge, force etc. The created solution space intends to spark creativity in a manner similar to checklists (compare chapter 5.5.5), but couples these systematic approaches with matrix-based techniques (compare chapter 5.5.6) and functional models (compare chapter 5.5.2 and the approaches discussed in chapters 6.1 to 6.4).

As a result, identified shortcomings and potentials based on the application of methods, as in chapter 6.4, can be addressed within the application of matrix-based analysis and modeling techniques. This broadens the solution space and points to new solutions, based on the analysis of existing functional and physical system representations.

The benefits of the approach include (compare Deubzer & Lindemann 2008):

[84] For an overview on different types of synthesis consult the introductory section of chapter 5.5.

- The possibility to work thoroughly and systematically through the solution space and explore possible solutions, as well as their impact on the functional and physical system representation;
- The possibility to correlate different layers of abstraction and analysis through coupling of e.g. states, operations, functions and physical parts;
- The identification of limitations and potentials, as well as their exchangeable solutions;
- Reasonable decision-making processes through achieving transparency of available solutions and their benefits or shortcomings.

6.6 Conclusion

The constituents to the approach discussed propose the use of the MDM approach for the modeling of the product architecture and the product architecture-related domains. Fundamental requirements were defined, such as the modeling in MDM notation, the coupling of methods and models, and the coping with hierarchies and recursive procedures. The methods were enhanced to suit not only analysis- but synthesis-purposes as well. The definition of conceptual solutions was shown, based on clear functional descriptions of existing technical systems. In the final example, focus was placed on the decoupling of a system into operations, states, and physical parts, to enable the application of networked models of working principles and the depiction of the solution space based on that representation.

The use of functional modeling in matrix notation allows for the definition of design rules to be conducted manually or automatically and for a structured comprehensive depiction of the solution space. Approaches such as clustering for modularization, path analysis and further structural characteristics can be applied within the overall solution space, enabling informed decisions in early phases of design. Analysis of the different solutions is enabled through the definition of user functions, properties of physical parts (such as weight, cost etc.), and operations (such as the degree of efficiency for example).

Radical innovations are supported by a fundamental and systematic variation of the system structure, as well as the possibility of adding new components (in the given example i.e. fuel cells or thermo-electric-converters etc.), thus expanding the solution space systematically.

Whereas the implied methods alone allow for a definition of discrete solutions, the presented approach supports the definition and depiction of the overall solution space, the visualization of solutions therein and the comparison of possible solutions. The MDM approach was enhanced by the cross-linking of different levels of abstraction, the composition of these levels, and the systematic navigation through them. The original MDM approach allows for the linking of domains as a snapshot of the current development situation, but not for the support of the establishment of models throughout the process, their composition and management.

From the perspective of variant management, the following advantages show (compare DEUBZER & LINDEMANN 2009c):

- The use of the matrix notation for variants enables an extremely high number of variants to be intuitively represented and processed.
- The representation of variant spectra with strength based graphs enables connections between variants to be recognized intuitively. Transparency is established, which, through its absence, is one of the main causes for the existing problems in handling variant diversity.
- The configuration rules, restrictions, and prohibitions, common when designing product portfolios, can be represented very efficiently in the matrix.
- The methods of cluster analysis enable the identification of core structures of variants and part numbers. This forms the basis for optimization of product programs. For instance, when the variants available in a product range are represented with the methodology, further opportunities for optimization show, as the variant matrix contains all possible feature combinations. The entire product program that is possible in theory can be derived by the analysis of the completely cross-linked clusters.

From the perspective of product architecture synthesis, the following advantages can be summed up:

- Systematic exploration of the solution space is enabled through the comprehensive depiction of the solution space, the coupling of domains and levels of abstraction, and the application of powerful analysis techniques.

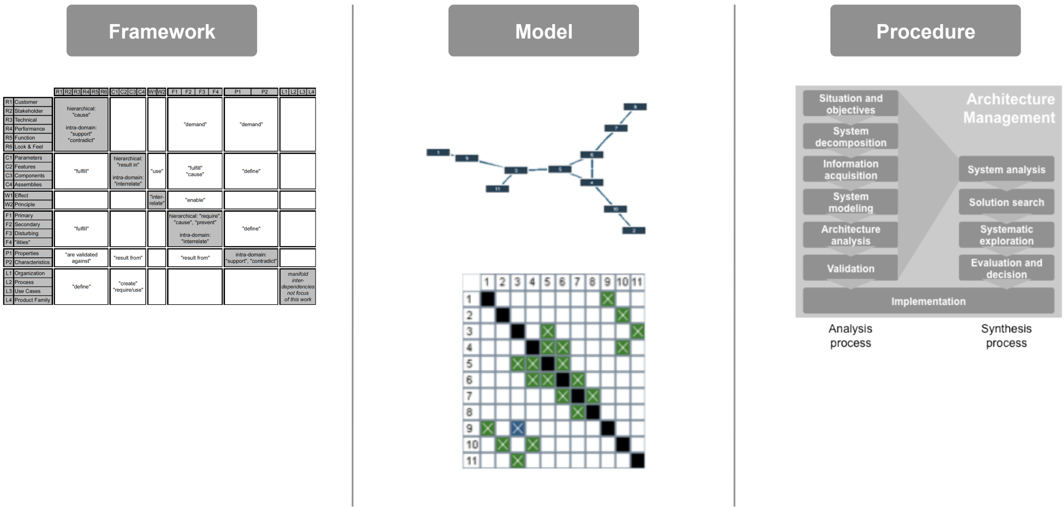
The evaluation of solutions based on the graph representation of the solution space enables the comparison of solutions from a structural point of view, since methods for structural analysis are applicable during the synthesis phase. The impact of changes and new solutions becomes transparent on the different levels of abstraction and in all different domains.

7. Solution approach: methodology to manage product architectures

The following chapter incorporates the previous discussions to provide a procedural model, based on the MDM methodology, which elaborates on the relevant steps for product architecture management, based on the tasks of systems architecting. The procedure is linked to the methods identified in the previous chapters, and, as such, provides an outline for the management of product architectures, supplying the relevant entities, relations and outcomes of the respective methods. The procedure is introduced together with the product architecture framework, providing the relevant entities, as well as an appropriate model. This approach will be applied to an example from the automotive industry in chapter 8.

7.1 Overview

The comprehensive approach for the management of product architectures consists of three core elements. The first pillar is a **framework**, depicting the different entities of product architectures and their interrelations (compare chapter 4.3). The second element is the **model** for product architectures, incorporating the discussions of the handling of product architectures (compare chapter 4.2). As a last constituent of the approach, a **procedural model** is defined, applying the framework and model. The procedure is not to be understood as sequential approach for the management of product architectures, but identifies relevant steps during the process, which can be chosen on demand, depending on the respective situation. Relevant cutouts of the framework and feasible methods accompany the steps of the procedural model (compare chapter 5).



	R1 R2 R3 R4 R5 R6	C1 C2 C3 C4	W1 W2	F1 F2 F3 F4	P1 P2	L1 L2 L3 L4
R1 Customer R2 Stakeholder R3 Technical R4 Performance R5 Function R6 Look & Feel	hierarchical: "cause" intra-domain: "support" "contradict"			"demand"	"demand"	
C1 Parameters C2 Features C3 Components C4 Assemblies	"fulfill"	hierarchical: "result in" intra-domain: "interrelate"	"use"	"fulfill" "cause"	"define"	
W1 Effect W2 Principle			"inter-relate"	"enable"		
F1 Primary F2 Secondary F3 Disturbing F4 "ilities"	"fulfill"			hierarchical: "require", "cause", "prevent" intra-domain: "interrelate"	"define"	
P1 Properties P2 Characteristics	"are validated against"	"result from"		"result from"	intra-domain: "support", "contradict"	
L1 Organization L2 Process L3 Use Cases L4 Product Family	"define"	"create" "require/use"				*manifold inter-dependencies not focus of this work*

Figure 7-1 Overview of the solution approach

The following chapters introduce and discuss the three elements of the approach in detail. The framework then builds on the entities of the product architecture identified in chapter 4.3, differentiated into the superordinate classes of goals, objects, and action. The product architecture model proposes the modeling technique with the capability of grasping the complexity of the product architecture framework and relies on the proposition of graph and matrix visualization (compare chapter 4.2). To complete the picture and enable the application of appropriate methods, the proposed procedure allows for the identification of situation-specific relevant cutouts of the framework and respective methods. The procedure is largely based on the processes of system analysis (compare chapter 5.4.1) and synthesis (compare chapter 5.5), while the aspects of functional modeling, evaluation, goal definition, etc. are largely part of – and thus incorporated into – these steps.

7.2 Framework for product architecture domains

The framework for product architecture modeling was introduced in chapter 4.3, including the most distinctive entities of the product architecture. The discussion of different methods for product architecture management in chapter 5 provides numerous linkages between those entities, based on the models and logical linkages of each method. The six major entity domains of the product architecture framework are the domains of **requirements**, **components**, the **working** and **functional** domains, **property** domain, and **lifecycle** domain. While the requirements and the lifecycle stand for themselves as the **system of goals** and the **system of action**, the domains of components, the working domain, functions, and the property domain are grouped as the **system of objects**.

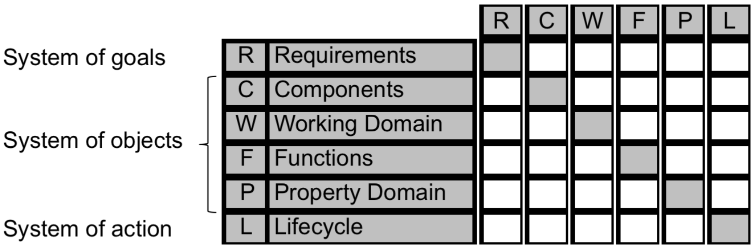


Figure 7-2 Major entity domains of the product architecture framework

Different methods combine those domains, such as QFD, which combines the components domain in terms of features with the requirements domain in terms of customer requirements (compare the section “Quality as evaluation criterion” in chapter 5.6.3). The domains are not only interrelated, but also detailed by a number of methods, such as a TRIZ-functional model, which provides linkages between primary, secondary, and harmful functions, all within the functions domain (compare the section “Relational functional model” in chapter 5.5.2). While this differentiation is made in the meta-model of the product architecture framework (see following full-page figure), other details are left out to allow for the framework to deliver a clear overview. The flow-oriented functional model, for example, separates functions into states and operations (compare the section “Flow-oriented functional model” in chapter 5.5.2), which are not depicted in such detail in the meta-model.

To complete the framework, the literature review of chapter 5 provides further linkages between the major entity domains, resulting in a meta-model as basis for product architecture management (see the following full-page figure). The meta-model shows the interrelations between product architecture entities, providing the outline for product architecture management activities. Since the meta-model provides only a rough outline, the numerous domains can be detailed to suitable granularity for each application, in relation to the respective step of the procedure. Not all entities are documented in the framework, to concentrate on the most relevant. The application of the model in chapter 8 will underline this practicability. It is important at this point to note, that particular projects require additional entities, which in turn need to be incorporated into the framework, depending on the situation. The following paragraphs discuss the major entity domains of the framework, the respective domains and the interrelations in detail. The meta-model exclusively depicts direct interdependencies, which is of great importance for the outcome of the application. Since indirect dependencies are often within the thoughts of the users, one or the other intuitive interdependency might be missing at first glance, yet implicitly available through indirect dependencies.[85] The meta-model is read, accordingly to the direction in the DSMs, as "row influences column".

The system of goals corresponds with the **requirements** as the major entity domain of the product architecture framework. The sub-domains are formulated within, classifying the occurring requirements depending on their source or category.[86] The sub-domains, according to the sources of requirements, are the *customer requirements* and requirements from other *stakeholders*. The product requirements are further divided into *technical*-, *performance*-, and *functional*-requirements, as well as requirements for the *look and feel* of the product (as an example for non-functional requirements).

The *inter-domain* dependencies focus largely on the domains of functions and properties. The requirements, of any sort whatsoever, demand a certain functionality or properties of the product architecture. While this is the general rule, exceptional cases might demand certain components or other entities of the component-domains, or certain physical effects or principles (if customers or stakeholders are closely involved with the technical solution, likely in business-to-business situations, for example). The hierarchical *intra-domain* dependencies among requirements can be characterized as "causal" relationships, i.e. customer or stakeholder requirements cause the existence of performance, functional or technical requirements. Within each domain of requirements, the requirements possibly support or contradict each other. While most of contradictions are caused by the technical solution and are thus indirect dependencies (which is often the case for customer requirements), requirements contradict or support each other directly as well, such as a maximum length might contradict the sum of other desired dimensions.

[85] Indirect dependencies can be visualized using domain-mapping logics, as discussed in the section on matrix-based analysis approaches in chapter 5.4.3.

[86] In comparison to the classification in chapter 5.3.5, focus was placed on product requirements (functional and non-functional), rather than process or organizational requirements (depicted as domains in the system of action), and the sources (here classified as customers and other stakeholders).

		R1 R2 R3 R4 R5 R6	C1 C2 C3 C4 C5	W1 W2	F1 F2 F3 F4	P1 P2	L1 L2 L3 L4
R1	Customer	hierarchical: "cause" intra-domain: "support" "contradict"			"demand"	"demand"	
R2	Stakeholder						
R3	Technical						
R4	Performance						
R5	Function						
R6	Look & Feel						
C1	Parameters	"fulfill"	hierarchical: "result in" intra-domain: "interrelate"	"use"	"fulfill" "cause"	"define"	
C2	Features						
C3	Components						
C4	Assemblies						
C5	Interfaces						
W1	Effect			"inter-relate"	"enable"		
W2	Principle						
F1	Primary	"fulfill"			hierarchical: "require", "cause", "prevent" intra-domain: "interrelate"	"define"	
F2	Secondary						
F3	Harmful						
F4	"ilities"						
P1	Properties	"are validated against"	"result from"		"result from"	intra-domain: "support", "contradict"	
P2	Characteristics						
L1	Organization	"define"	"create" "require/use"				*numerous inter-dependencies; not focus of this work*
L2	Process						
L3	Use Cases						
L4	Product Family						

Figure 7-3 Meta-model of the product architecture framework

The entities of the domain of **components** can be grouped into sub-domains as well. *Parameters* and *features* constitute the smallest entities of the sub-domains. *Components* and *assemblies* are the larger physical entities of the components domain. For the generic framework, assemblies were chosen as a grouping of components. In projects and applications of the model, the hierarchies of physical entity domains often include a larger number of levels (sub-assemblies, assemblies, modules, main-modules, products, etc.). Since the principle behind the coupling remains the same, the components and universal assemblies were chosen to clarify the composition of the domain.

As an *inter-domain* dependency, components fulfill requirements and functions (for example stiffness, corrosion etc.), and thus are linked to the requirements domain. Furthermore, since harmful functions are also part of the framework, the link-type “cause” is also valid for the interrelation of components and functions. In most cases, functions of the product architecture are not fulfilled by a single entity of the component domain, but by a number of entities. Components, and especially assemblies, make use of physical effects and working principles, while defining properties of the product architecture on the other hand. Again, the properties and characteristics of the product architecture are likely to be defined by a number of entities (compare structural characteristics in chapter 5.4.3) rather than one lone entity.

The composition of sub-domains already indicated the major *intra-domain* dependency type of the components domain. Hierarchically, the smaller entities result in larger entities, such as features result in components and components result in assemblies. Within each domain of components, the dependency type was – admittedly vaguely – defined as “interrelate”. As was discussed in previous chapters (compare 5.4.3), the dependencies between components can be characterized as spatial, functional, material, energy, etc. At this point, it has to be clarified that especially spatial dependencies have to be considered as direct dependencies, which can hardly be deducted from other domains, while functional dependencies, material or energy flows might as well stem from the functional domains (compare chapters 5.5.2 or 6.4). Accordingly, the mentioned and further dependency types might apply respectively to the project and use case. In addition, the interrelations between components can be defined by interfaces, which then pose as an element between components.

The **working domain** provides the entity domains of *physical effects* and *working principles*. Both bridge the gap between the components and the functional domains by providing principal solutions to desired functions. Consequently, functional and component domains represent the major interdependencies of working principles and physical effects.

While being used by components, the working domain is related by *inter-domain* dependencies to functions as an enabler. To fulfill a function, working principles and physical effects are required to provide the functionality in combination. This is an important notion, since working principles seldom provide functionality alone, especially in the context of complex product architectures. As was discussed in chapter 5.5.5, principles and effects pose an important step during the synthesis process, concretizing functions to components.

Within their domain, working principles and physical effects also interrelate. Again, the *intra-domain* dependencies are not further specified. The interrelation regularly focuses on the input and output parameters of effects, but might also depend on the scale of the effects, e.g. the amplitude of force provided as an input- and output variable.

The domain of **functions** was intensively discussed in chapter 5.5.2, introducing numerous functional models, their advantages and disadvantages. For the product architecture framework, all relevant domains of functions were considered. *Primary functions* describe the main functions of the product architecture or its main purposes. The supporting *secondary functions* are required for the primary functions to take effect. As within the component domains, more levels of hierarchy may apply for different projects, as the examples of hierarchical functional modeling in chapter 5.5.2 have shown. *Harmful functions* are a major achievement of the relational functional model. The introduction of harmful functions allows for the identification of negative side effects as a result of desired functionality, the identification of conflicts of objectives, and the respective sources of these effects. Since not all of the qualities of the product architecture can be grasped by the introduced entities, the domain of *"ilities"* is part of the product architecture framework, where the overall qualities such as manufacturability, recyclability, etc. are included, to fully characterize the product's functionality.

Aside from the *inter-domain* dependencies discussed in the previous sections, the functional domains relate to other domains in the following ways: where requirements demand a product's functionality, the resulting functions intend to fulfill the formulated (functional) requirements. On the other hand, functions play their role in defining the product properties. While components define properties, such as weight and other physical properties, functions define the functional properties, both positive and negative, of the product architecture.

The *intra-domain* dependencies were intensively discussed within the context of functional modeling. The hierarchical dependencies include the dependencies between the introduced sub-domains, such as: primary functions require secondary functions; different functions possibly cause harmful functions; or different functions are introduced to prevent harmful functions.[87] Since the functional models are heterogeneous in character, the dependencies within each domain depend on the chosen dependency logic. Typical dependencies are energy-, force-, information-, or signal-flows in the flow-oriented functional model, for example. Other dependencies result from indirect dependencies, derived from the dependencies of functions fulfilling components, requirements, etc.

The domains of product **properties** are strongly connected with the domains previously defined, and conclude the system of objects. While properties are demanded by requirements on the one hand, they are also defined by components and functions. Two sub-domains exist within the product architecture framework. The first domain is provided by *properties*, i.e. the general qualities of the product architecture, encompassing properties resulting from components and functions. The product architecture *characteristics* describe the rather architecture-specific properties, mainly represented by structural characteristics, as described in chapter 5.4.3, resulting above all from the structural dependencies of the product architecture entities. In general, properties need to be differentiated into desired and undesired properties, while especially undesired properties result from the interplay of numerous entities of the component and functional domains.

[87] For a detailed introduction see the examples of functional modeling in chapter 5.5.2.

While requirements propose desired properties, the resulting or actual properties possess *inter-domain* dependencies to the requirements, in the sense of validating the product properties. The validation of properties against requirements is an important part of the framework, reflecting essential parts of the presented procedural model and other previously discussed procedural models.[88] Product properties and characteristics result from components, functions, and the coupling of the numerous entities. In an ideal case, the resulting properties reflect the desired properties, yet undesired properties occur frequently, especially in complex systems. The interrelationships between properties on the one side and components and functions on the other are deliberately documented in both directions. It is thus made clear that the components and functions are designed to achieve desired properties, yet all properties, whether desired or undesired, result from the interplay of architecture entities, such as components and functions.

The *intra-domain* dependencies of the property domains behave similarly to the requirements. Properties support or contradict one another, and are largely dependent on the indirect relations resulting from the realization by physical components. Nevertheless, desired and undesired properties contradict each other, for example material which highly stable as a desired property but also brittle as undesired property. Such conflicts of goals have to be solved, if possible, and allow for the comparison of solutions based on the desired and undesired properties.

After the system of objects, the system of action represents the last group of domains within the product architecture framework, entitled **lifecycle** domains. The domains within the lifecycle group are heterogeneous and provide boundary conditions for the analysis and synthesis of product architectures. The proposed sub-domains of the lifecycle group are the organization, process, use cases, and the product family. Naturally, for each of these, an individual framework could be established, but is not focus of this work.[89] The presence and incorporation of the domains stress the attention for the inherent concerns and constraints for the product architecture. The domain *organization* provides information about the company organization structure and is relevant to allocate stakeholders, concerns and viewpoints. The *process* domain provides information about the process architecture, and can be detailed to the domains of relevant tasks, information and data flows etc. The definition of *use cases* may vary, from depicting the product lifecycle and the interaction of users with the product to the use of modules within the product family. The *product family* itself can be depicted by a number of entity types, such as the product lines, product types, platforms, modules, etc.

The entities within the system of action or lifecycle domain inherit *inter-domain* dependencies, mainly for requirements and components. The organization and its stakeholders, the company's process architecture as well as the product family or numerous use cases, define requirements of the product. Requirements stemming from the product family can be described as boundary conditions, since parallel product lines, platform and module strategies etc. predefine parts of the architecture. Use cases enable the derivation of

[88] Different models were discussed in chapter 1.1.3 on engineering design processes.

[89] Compare KREIMEYER [KREIMEYER 2010].

requirements. The depiction of the product lifecycle and use cases of different stakeholders therein (manufacturer, logistics, customer, disposal, etc.) is a common method for the definition of requirements. Stakeholders within the organization and existing business and production processes may define requirements directly, at least allowing for the derivation of requirements. Production sites especially usually define requirements of different types, such as the limitations of the number of units, size of the products, etc. While the interdependencies to the domains of requirements are largely homogeneous, the dependencies between the lifecycle domains and components are heterogeneous, as are the domains themselves. To define the interdependencies clearly, a differentiation is made between the two domains of organization and process on the one hand, and the domains of use cases and product family on the other. The organization and process define aspects of the product architecture indirectly, through the definition of requirements, which again demand functions and properties. From the perspective of organization and processes, the business processes and the underlying organization, such as engineering and production as common examples, create the components. The product family and use cases require or use the physical entities of the component domain. Use cases rely on the physical product to depict the usage during its lifecycle. The product family incorporates all physical entities and their numerous interrelations, i.e. commercial, technical, functional inter- and intra-domain dependencies etc., with the goal of offering a cost-efficient and comprehensive product portfolio.

With the lifecycle domains providing the boundary conditions for the product architecture, the *intra-domain* dependencies of the lifecycle domains are a subordinate concern of the product architecture framework. The interdependencies within the process domain alone are numerous, considering the numerous classes of entities, such as tasks, people, information etc. As such, the intra-domain dependencies of the lifecycle domains are not focus of this work and are not further discussed.

The following sections will clarify in detail how to model the architecture framework and to systematically generate the content of the model. The structured procedural model enables the goal-oriented handling of the overall product architecture framework.

7.3 Model for product architectures

To model the entities of the product architecture framework, the core idea for a product architecture model was introduced in chapter 4.3, based on the requirements for the modeling of product architectures presented in chapter 4.2. The proposed modeling outline, based on graphs and matrices, can be detailed with the information on the architecture framework introduced in the previous chapter. The following sections will discuss the modeling approach and introduce in detail how to cope with the model, based on the MDM approach.[90]

The multiple-domain mapping approach aims to depict, analyze and handle complex systems, which are characterized by numerous (types of) domains, entities and interrelations. Within the product architecture framework, the domains are grouped by an overall classification (system of goals, objects, and action), and classes of domains (i.e. requirements, components,

[90] For a detailed overview and introduction to the MDM approach see MAURER [MAURER 2007].

functions, etc.). The product architecture framework introduced the dependencies on the level of classes of domains. The dependencies within were differentiated into hierarchical intra-domain dependencies (e.g. dependencies between domains of the same class of domains, such as domains of the class requirements), intra-domain dependencies between entities in general (e.g. customer requirements), and inter-domain dependencies, e.g. between the domains of requirements and the domains of functions. The following figure depicts the composition of the framework.

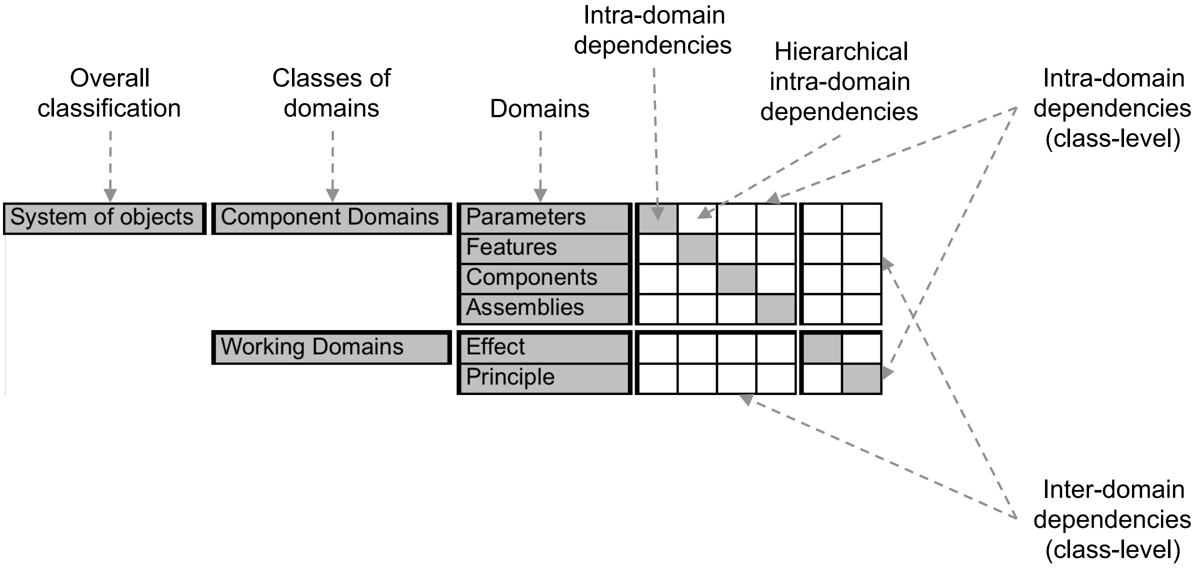
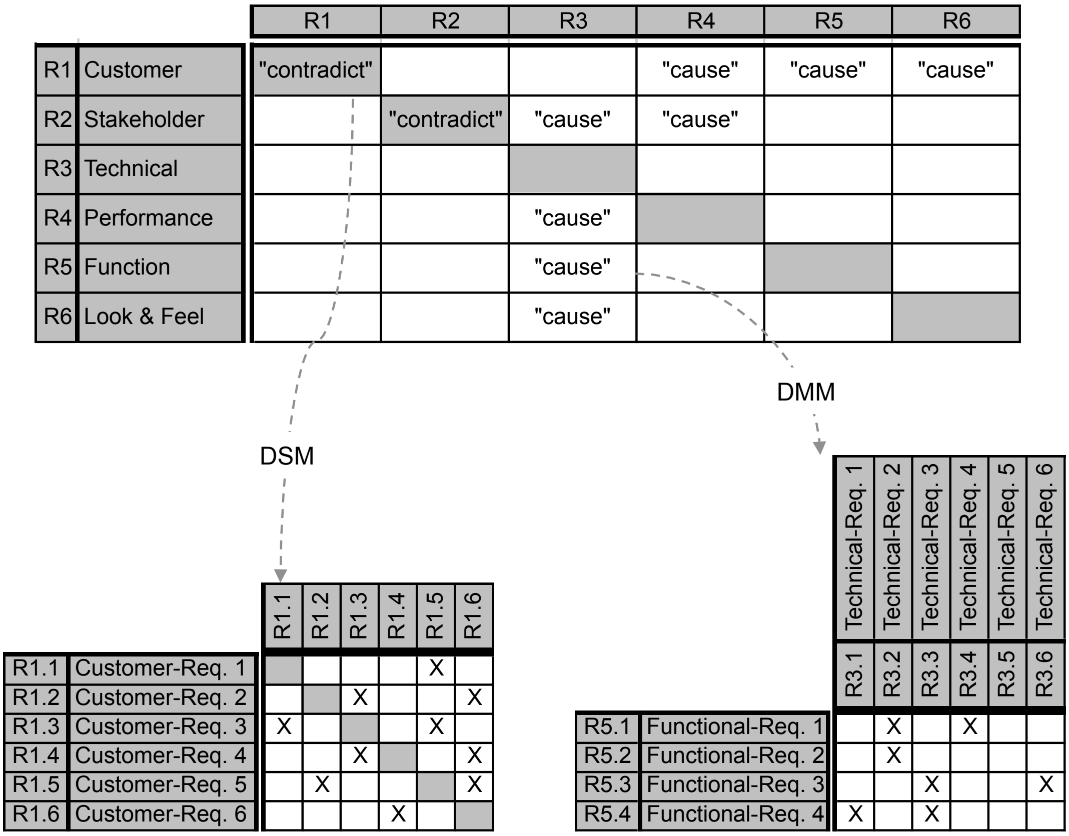


Figure 7-4 Basic composition of the framework and model

The basic composition of the product architecture framework provides the meta-model for the MDM application, i.e. the domains and types of interrelationships between domains. The interdependencies provided therein can be detailed to the individual domains and their respective interdependencies, down to the dependencies between individual entities. The following figure gives an example of this. Whereas within the framework, "cause", "contradict", and "support" are identified as the possible intra-domain dependencies between requirements, the three possibilities can be detailed down to the respective domains, i.e. customer- and stakeholder-requirements etc. (see top of the following figure). Within the class of domains, the interdependencies are differentiated again into inter-domain (i.e. hierarchical intra-domain) dependencies and intra-domain dependencies. Inter-domain dependencies are depicted within a domain-mapping matrix (DMM) and intra-domain dependencies within a design structure matrix (DSM). At the bottom of the following figure, the DSM shows the interdependencies between customer requirements, all of the type "contradict"[91], while the DMM depicts the interdependencies between functional and technical requirements of the type "cause".

[91] Such a DSM is part of the application of the method QFD, for example (compare chapter 5.6.3 on quality).

With the information given in the model, the structural analysis and synthesis methods discussed in previous chapters can be applied, including domain-mapping logics. Domain-mapping logics allow for the derivation and translation of dependencies between domains, and thus the differentiation between direct and indirect dependencies.



		R1	R2	R3	R4	R5	R6
R1	Customer	"contradict"			"cause"	"cause"	"cause"
R2	Stakeholder		"contradict"	"cause"	"cause"		
R3	Technical			▨			
R4	Performance			"cause"	▨		
R5	Function			"cause"		▨	
R6	Look & Feel			"cause"			▨

		R1.1	R1.2	R1.3	R1.4	R1.5	R1.6
R1.1	Customer-Req. 1	▨				X	
R1.2	Customer-Req. 2		▨	X			X
R1.3	Customer-Req. 3	X		▨		X	
R1.4	Customer-Req. 4			X	▨		X
R1.5	Customer-Req. 5		X			▨	X
R1.6	Customer-Req. 6				X		▨

		Technical-Req. 1	Technical-Req. 2	Technical-Req. 3	Technical-Req. 4	Technical-Req. 5	Technical-Req. 6
		R3.1	R3.2	R3.3	R3.4	R3.5	R3.6
R5.1	Functional-Req. 1		X		X		
R5.2	Functional-Req. 2		X				
R5.3	Functional-Req. 3			X			X
R5.4	Functional-Req. 4	X		X			

Figure 7-5 Project-specific decomposition of the product architecture framework down to entity-level

The matrices established within the model are to read as “row influences column”, wherein different dependency types can be differentiated and weightings of dependencies applied (compare the discussions on matrix-based approaches in chapters 5.4.3 and 5.5.6). As the reference to QFD in the above example has shown, existing models can be used as input for the product architecture model (compare chapter 6.1), and the different methods and models can be interrelated within the product architecture model (compare chapter 6.2).

The corresponding matrix and graph depiction of the same system was brought up by MAURER [MAURER 2007] and already introduced and discussed in chapters 4.3, 6.3, and 6.5. The graph depiction in particular turns out to be intuitive and accessible for users, while the matrix depiction is superior in terms of a systematic approach to complex systems. Both visualization forms provide mathematical accessibility, since the inherent information about the system is the same. Comprehensive analysis approaches can be applied due to the mentioned possibilities.

Depending on the analysis results, they can be more easily displayed in matrix or in graph representation, depending on the type of characteristics to be displayed. In many cases, characteristics involving a large number of entities and interrelations are easier to intuitively grasp using graph depiction (such as paths or cause-and-effect chains, clusters, feedback loops etc.). On the other hand, characteristics concentrating on one element, be it an entity or interdependency, are often easier to understand employing the matrix (such as the active- and passive sum of an entity, identification of bus-elements etc.).

To conclude the product architecture model, the requirements in chapter 4.2 can be met by the proposed modeling approach, and the framework for product architecture management can be depicted by the product architecture model. The following sections will introduce a procedural model, which in turn aims at coping with both the framework and the model.

7.4 Procedure and methods for product architecture management

The procedural model and mapping of methods for product architecture management is intended to guide the application of the discussed framework and model of the product architecture. Depending on the project, goals and objectives, the emphasis on the different steps may differ. The procedural model is not intended to either depict the engineering design process or replace existing problem solving procedures (compare chapter 1.1.3). Nevertheless, the provided procedural model supports the conduction of engineering design processes and/or problem solving procedures. The specific aspects of the product architecture are represented by the procedure. Coupling the procedure to engineering design projects helps to support the incorporation of architectural aspects into the projects.

7.4.1 Interpretation of the procedural model

The procedural model is largely based on the discussions presented in chapter 5 regarding the steps of the procedure, as well as the integrated methods. The methods were intensively discussed in chapter 5, and their applicability for product architectures was also assessed. The differentiation into the system of goals, objects and action was chosen for the structuring of methods, while the procedure presented in this chapter is founded on the core activities, which represent the design of product architectures, i.e. the **interplay of analysis and synthesis**.

As such, the procedural model combines those two major streams into **one coherent model** (see figure at the end of this section). Further topics discussed in chapter 5, such as the goals and requirements, evaluation, and lifecycle perspectives, are part of the two major streams, for example requirements in the step “situation and objectives” or evaluation as part of the step “evaluation and decision”. The potential of the combination of analysis and synthesis lies within the coupling of powerful analysis methods with those of synthesis, all **based on one product architecture model**.

Clearly, the depicted flow-orientation of the procedural model can only serve as a rough outline. As discussed in chapter 1.1.3, the engineering design process is as much an iterative and recursive procedure as the process of product architecture management process. Therefore, any procedural model can only insufficiently describe the actual chronological sequence of steps in different projects. The procedural model presented therefore aims to **provide distinct steps, each with distinct goals**, whose sequence and emphasis can be formed by the user. While the presented sequence is based on scientific research, different sequences might be useful for different projects. Although that the **iterations and recursive procedures** are not explicitly displayed, they are still an inherent ingredient of the philosophy of the model.

The process of analysis is naturally considered to be the first step of synthesis (compare chapter 5.5.1), while synthesis might be considered as part of the implementation process during analysis (compare chapter 5.4.1). Within the presented model, implementation is considered as both: it can be either the preparation of synthesis after analysis, or the implementation of solutions after successful recommendations from the synthesis process. Implementation itself is not the central focus point of the model, but the inherent recommendations and reasonable interpretation of analysis- and synthesis-results are.

The following figure displays an overview of the procedural model. The steps of the analysis process follow the definitions and explanations of chapter 5.4.1, which is also valid for the synthesis process and chapter 5.5.1. The following section discusses each step individually, pointing out the goals, methods, models, and critical aspects of each step, as well as the architecture entities of primary consideration. The steps follow the processes of analysis and synthesis sequentially, as is depicted in following figure.

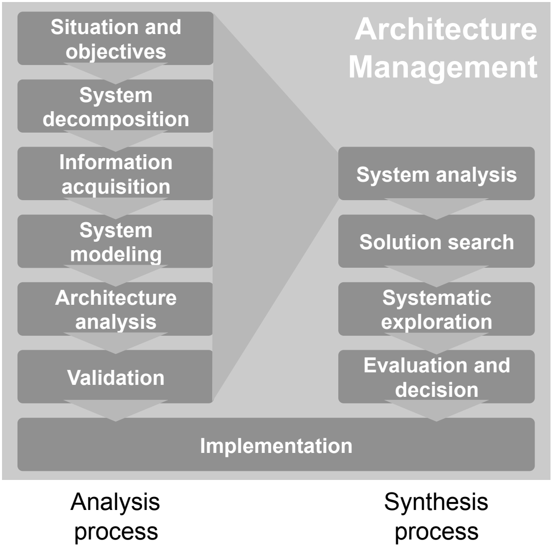


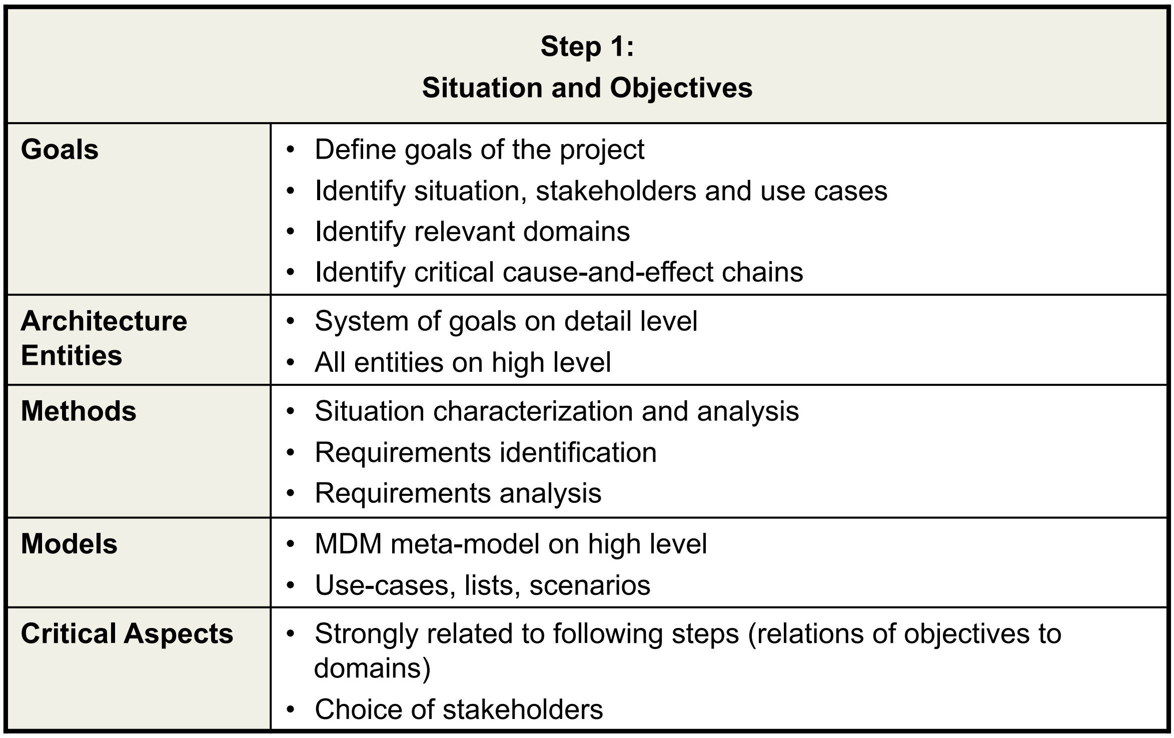
Figure 7-6 Procedural model for the management of product architectures (iterative and recursive procedures not explicitly displayed)

7.4.2 Steps and methods of the procedural model

To clarify intentions and focal points of the steps of the procedure, the following sections discuss each step individually, pointing out the key factors to be taken into account. In order to not repeat previous elaborated details, the discussions in the following paragraphs will reference previous chapters. The procedural model intends to give an overview of how to cope with the product architecture model and framework. Suitable methods are indicated through references to the previous chapters. The introduced constituents of the approach (chapter 6) are aligned with the methods identified from the state of the art (chapter 5). The combination allows for a comprehensive solution to the management of product architectures.

Goals and objectives

The definition of goals and objectives as a typical first step of the project is very crucial, while still being required to be questioned and validated throughout the runtime of the project. In general, the goals of projects in the context of product architecture management can vary greatly. Projects with the focus of analysis of existing architectures primarily ask why a certain behavior or property occurs, what the causes and/or effects of certain entities are, which boundary conditions or degrees of freedom for synthesis exist etc. Synthesis projects question how requirements or desired properties can be reached, which possible solutions exist, how decisions between concepts can be reached etc. Other areas of focus, variant management, for example, might combine the two, i.e. analyzing the existing portfolio on the one hand, while seeking solutions based on the identified boundary conditions and restrictions. For those types of projects, the comprehensive approach with continuous modeling and method application is even more crucial.



Step 1: **Situation and Objectives**	
Goals	• Define goals of the project • Identify situation, stakeholders and use cases • Identify relevant domains • Identify critical cause-and-effect chains
Architecture Entities	• System of goals on detail level • All entities on high level
Methods	• Situation characterization and analysis • Requirements identification • Requirements analysis
Models	• MDM meta-model on high level • Use-cases, lists, scenarios
Critical Aspects	• Strongly related to following steps (relations of objectives to domains) • Choice of stakeholders

Figure 7-7 Procedural model for the management of product architectures: Step 1 "Situation and objectives" – overview

To be able to define the goals and objectives of the project, it is the primary goal of this step to grasp the current situation, the relevant stakeholders, and the existing use cases, if necessary. For a systematic grasping of the situation, PONN established a descriptive model for development situations in the context of target-oriented method selection. Direct context (actual and desired situation) within the model is differentiated from indirect context (influencing factors such as people, boundary conditions, tasks, and the product) [PONN 2007, p. 123]. Different influence factors for the situation support the adaptation and application of the descriptive model [PONN 2007, pp. 50 ff.].

To support the situation definition, the identification of stakeholders and use cases for each stakeholder group is a powerful measure. Stakeholders can be customers using the product, workers assembling the product, decision makers for buying products etc. Essentially, all groups involved along the product lifecycle can be considered as stakeholders (compare chapter 5.3). To identify objectives and goals for each stakeholder group, the modeling of use cases for each group is applied. The use cases describe the activities each stakeholder is conducting with the system, from which requirements for the solution or objectives for the project (for example high level requirements for the solution) can be derived. The use cases can be considered as application- or use-scenarios for each stakeholder group.

The identification of stakeholders as a first step describes the detailing of one domain of the system of goals, considering the product architecture framework as a starting point. In the following, all domains that are relevant for the considered problem can be identified, based on the identification of stakeholders and use cases. The product architecture framework provided in chapter 7.2 is intended to support this process by providing elementary domains of the product architecture and their interrelations, yet does not claim to be complete or sufficient for any problem or project.

It is important to note that it is not the system decomposition that is sought at this point, but rather the identification of domains and interrelations between domains, based on the architecture framework provided. The resulting meta-model of the architecture, depicting the relevant use cases and domains, provides the basis for the following steps of system decomposition and information acquisition. These are closely related to one another, together with the plausibility check and validation of the meta-model defined in this first step of goals and objectives.

To document the hypotheses defined at the beginning of the project and grasp the most critical domains and relationships, the identification of critical cause-and-effect chains concludes the main activities of the first step. The critical cause-and-effect chains can be documented based on the domains and their interrelations. For example, the customer demands functional or performance requirements, which necessitate that the system provide certain functionality. The functionality is based on the realization through components, which causes harmful functions, resulting in undesired properties. The undesired properties cannot be directly tracked and eliminated, since they occur due to the interplay of all of the parts combined, thus demanding a thorough system analysis developed around the described primary cause-and-effect chain in terms of relevant domains and interrelations. Again, at this point, it is not desired to identify each customer requirement, component, and function involved, but to understand the core of the occurring challenge and set the framework for thorough system analysis in the following steps.

The architecture entities considered at this early point, as described, focus on the system of goals. While requirements and stakeholders to be defined are rather detailed, further domains, such as those of the system of objects and action, are described solely on domain-level. This procedure prepares the following steps of system decomposition and information acquisition.

The methods chosen for this step are those of situation analysis in general, as well as use case-based approaches, as described in the above sections in the context of the step's goals. If a highly product-related or typical development project is chosen, the first step described is strongly related to the typical requirements engineering approach, accompanied by the more far-reaching domain-related framework approach. As such, the principles of requirements identification and analysis can be applied during this step (compare chapters 5.3.1 and 5.3.2).

The models available for a step, as described, are mainly represented by a high level system representation based on an MDM meta-model, i.e. the relevant domains and their interrelations. To support the definition of the meta-model, which provides the basis for the following activities, further available models, such as use case diagrams, requirements lists, scenarios etc., can be used.

The critical aspects of the step were already indicated as crucial activities in above sections. Above all, the close interdependence between the definition of objectives and the following steps of system decomposition and information acquisition was stressed. To conduct these steps sequentially is seldom the right approach, since for example the system decomposition might reveal further required domains, which again need to be integrated into the framework etc. The choice of stakeholders and related use cases is equally critical, since the objectives and interrelated domains are largely based on the perception of the actual situation. Modeling per se requires neglecting parts of the reality, accompanied by the downsides of modeling, examples of which were discussed in chapter 4.2.

System decomposition

The definition of goals and objectives set the framework for the system decomposition. While the system of objects was considered in detail, the remaining domains, belonging to the systems of objects and action, were only defined at a high level.

System decomposition now strives for the detailing of these domains in two directions. The first direction describes the identification of existing hierarchical layers within each domain, i.e. the typical approach for system decomposition.[92] The decomposition might, for example, detail stakeholders according to organizational structures, functions or components on different modular levels, etc. The second direction of system decomposition aims for the detailing of the interrelations between domains, based on the given framework. Following the example of the step goals and objectives, the customer requirements can be detailed, displaying the cause-and-effect relationships among functional requirements etc. The functional view of the product architecture can be detailed in many ways (compare chapter 5.5.2), enabling the linkage of customer requirements to the functional model on the required level of detail and viewpoint. Following the train of thought of the example, the modular

[92] Chapter 5.4.2 discusses hierarchies in general, while chapter 5.5.2 elaborates the concept of hierarchical functional modeling. The downsides mentioned in those sections can be partially overcome by the concepts provided in chapter 6.3.

decomposition of the product can be conducted as well, pointing out which module(s) are involved in the function fulfillment and how they are embedded in the system.

Step 2: **System Decomposition**	
Goals	• Identify levels of abstraction within domains • Concretize interrelations between domains on identified levels of abstraction
Architecture Entities	• All entities on high level • Focus on objective-related domains • Concretization in all domains and levels of abstraction, based on examples
Methods	• Principles of analysis • Detailed use cases
Models	• Hierarchical system representation of each domain • Refined MDM meta-model
Critical Aspects	• Strongly related to previous step • Basis for information acquisition

Figure 7-8 Procedural model for the management of product architectures: Step 2 "System decomposition" – overview

The decomposition thus allows for not only the allocation of entities on different layers of abstraction, but also the identification of interdependencies within and across different layers of abstraction. As a result, information acquisition and the following processes of analysis and synthesis can be conducted on the required level of detail. The following steps, e.g. a detailed functional modeling of the system, might nevertheless require the consideration of more than one hierarchical layers, yet the discussion and information acquisition can be conducted explicitly and guided within each layer. The same is valid for the solution finding on different layers and domains of the system of objects etc. A number of examples are sufficient to identify the differentiated layers of each domain. Again, iterations and recursions among the steps of the procedural model have to be taken into account, not only as unwanted side effects, but also as deliberate property of the model.

The entities to be extensively discussed and portrayed within this step are naturally all high-level domains that were identified and discussed within the step of goals and objectives. The spanning of the system across hierarchical layers, for example, allows for the framework to be detailed in accordance with the given objectives, focusing on representative examples within this step.

To systematically accomplish the system decomposition, the principles of analysis, as discussed in chapter 5.4.2 can be used to abstract, select, scale, encapsulate, etc. the system and its parts. The detailing of use cases supports the system decomposition by providing examples of comprehensive context. Further measures include available and standardized methods already describing the system at the project beginning; the intensive analysis of these documents and models is conducted within the step of information acquisition.

The system representation desired at this point provides the hierarchical layers of each domain, including the interrelations within and across domains and layers of abstraction, represented in graph and matrix form by the product architecture model. The status can be described as a refined meta-model of the product architecture.

As was discussed in the previous step, the most critical aspect of this step is the strong interrelation with previous and following steps. Again, the procedure has to be considered highly iterative and recursive, especially among the first three steps described.

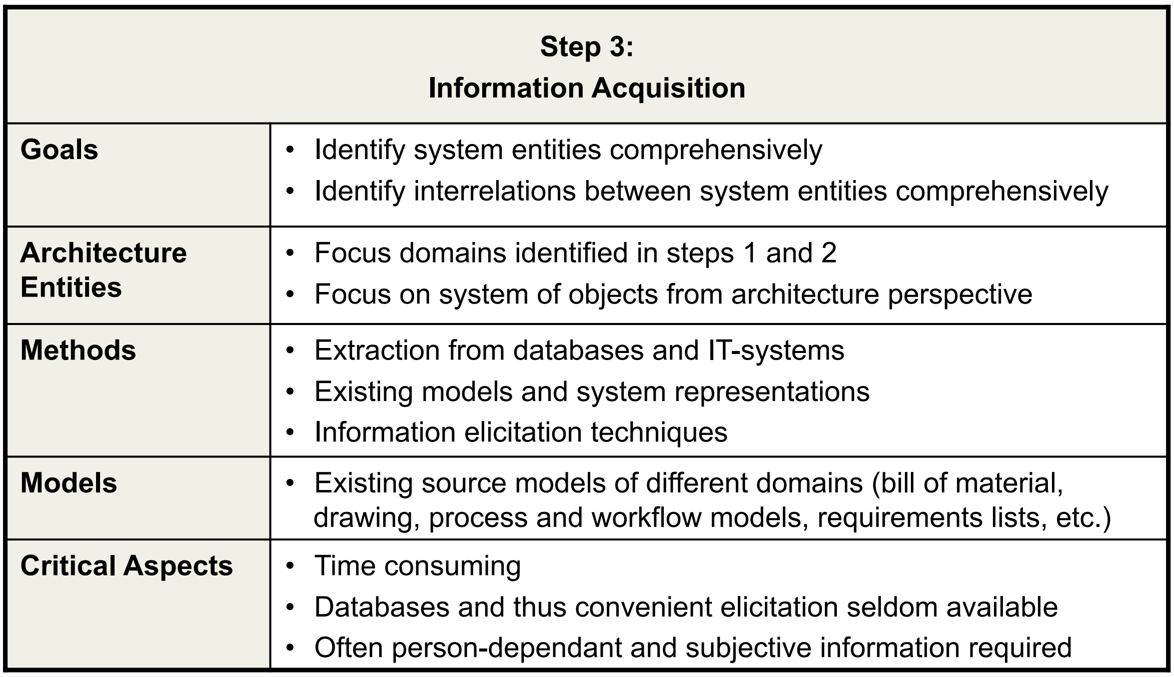
Information acquisition

Information acquisition is a highly complex and time-consuming activity in any analysis and synthesis project. Within this work, the process of information acquisition is not as intensively discussed as would be necessary to fulfill the importance of the step itself. Given the requirements for a solution to product architecture management (compare chapters 1.2 and 5.8), the focus is placed on the utilization, transfer, and interrelation of existing models. A brief discussion of the topic is part of chapter 5.4.1 in the context of system analysis, while the task of information acquisition is equally required for the processes of synthesis and evaluation, for example.

The goal of the step of information acquisition is to complete the product architecture framework, up to now available as detailed meta-model. The completion includes the identification of system entities (i.e. every singular relevant entity of the system) and the identification of system interrelations (i.e. every singular relevant interrelation of the system). The previous steps established the basis for making the right choice, in terms of both domains and hierarchical layers.

As a consequence, the relevant entities and domains of the step reflect those identified in the previous steps. The step of information acquisition might reveal further necessary domains and/or hierarchical layers, which have to be included in the meta-model as well. From the perspective of the product architecture, to achieve a sound understanding of the system, the focus is usually placed on the system of objects during the step of information acquisition, while the systems of goals and action are equally relevant in cases of requirements management or variant management projects, for example.

The methods to gain information can be divided roughly into three groups. Ideally, yet in the fewest cases, the information is available in existing databases or other IT-systems, which allows for a direct export into neutral data-formats accessible to analysis. Aside from the possibility of relying on automated and digitalized measures, other existing models and system representations can be used; for example, functional models, modular structures, results of FMEA- or QFD-application etc. Often, neither digital nor analog product representation is available completely, making the information elicitation from individuals necessary. Being the least reliable method, due to subjective and situational influences, highly systematic and professionally moderated approaches are required, for which techniques were discussed in chapter 5.4.1. Additionally, experienced moderating factors are inevitable.



Step 3: Information Acquisition	
Goals	• Identify system entities comprehensively • Identify interrelations between system entities comprehensively
Architecture Entities	• Focus domains identified in steps 1 and 2 • Focus on system of objects from architecture perspective
Methods	• Extraction from databases and IT-systems • Existing models and system representations • Information elicitation techniques
Models	• Existing source models of different domains (bill of material, drawing, process and workflow models, requirements lists, etc.)
Critical Aspects	• Time consuming • Databases and thus convenient elicitation seldom available • Often person-dependant and subjective information required

Figure 7-9 Procedural model for the management of product architectures: Step 3 "Information acquisition" – overview

In the presented step, the information is detailed towards the meta-model, resulting in suitable representations of the system (e.g. functional models for functions, modular structures for components etc.). For the product architecture management approach, the product architecture model was proposed in matrix- and graph form, capturing gathered information to establish the product architecture model and making information available for the following steps. Existing source models provide the basis for information acquisition, as was discussed in the previous section.

A number of critical aspects for this step can be identified, besides the often-mentioned, interwoven and iterative character of the first three steps. Since databases or other digitalized formats are seldom available, the process of information elicitation turns out to be immensely time-consuming. The preparation and conduction of workshops, for example, requires large amounts of human resources to be successful. The choice of individuals to participate is equally crucial, since information is often subjectively influenced and person-dependent. The elicitation of information from non-digital models can be considered as time-consuming and subjective as the gathering of information in workshops and interviews.

System modeling

The system modeling step can be described briefly, since it provides the information that was structured and gathered in the previous steps into one comprehensive model in MDM notation.

Step 4: **System Modeling**	
Goals	• Establish comprehensive model in MDM-notation • Preparation and planning of analysis (based upon focused entities)
Architecture Entities	• Focus on goals and object system • Action system with second priority
Methods	• Model transfer to MDM
Models	• MDM model
Critical Aspects	• Comprehensive information required • Sufficient levels of abstraction • Information to be transferable to MDM notation • Only correct input enables reasonable results

Figure 7-10 Procedural model for the management of product architectures: Step 4 "System Modeling" – overview

Accordingly, the primary goal of the step is the establishment of such a model. Based on the model and the project objectives, the preparation and planning of analysis is to be conducted, i.e. the identification of relevant domains, reasonable domain-mapping approaches and means of analysis, such as structural characteristics (compare chapter 5.4.3).

Naturally, the system modeling is conducted with a focus on the systems of goals and objectives, as were the previous steps. Since restrictions and dependencies might occur due to process- or product family limitations, the system of action is considered as well, though with less priority.

As a supporting method, the modeling in MDM notation is conducted according to the procedure described in chapters 6.1 and 6.2. Based on the procedure described there, existing models can be transferred to MDM notation easily and efficiently, resulting in a comprehensive model, where source models with overlapping entities are coupled accordingly.

The resulting MDM model provides the basis for the following steps of analysis and synthesis. Further information, which was not gathered on the basis of existing source models and methods, can be implemented in graph and/or matrix form directly into the MDM model.

Given the approach presented for system modeling, a number of critical aspects must be considered in order to provide a reliable and comprehensive model for analysis and synthesis. First of all, comprehensive information is required to provide representative results. As this information is not naturally available in digital or non-digitalized form in most cases, there is a probability that information will be neglected, providing invalid results. The neglecting of information can occur on all levels of the approach, i.e. ignoring relevant domains, hierarchical layers, interrelationships, or single entities or interrelations. Furthermore, the identified hierarchical layers are required to be on adequate levels for the desired purpose. Identified problems and challenges, which cause the project in the first place, need to be addressed on the basis of the defined model. Again, iterations might occur due to insufficient results, based on the analysis and validation results. As a next crucial element of this step, the gathered information needs to be transferable to MDM notation. The model is designed to suit the demands in early phases of development, thus focusing on qualitative, at times fuzzy, information. As for any method, a reasonable outcome and sufficient results can only be achieved by the provision of correct input information, underlining the importance of information acquisition. The validation step should provide the means to evaluate the analysis results and enable the assessment of the quality of information.

Architecture analysis

The step of architecture analysis represents the core step of the analysis process, together with the step of implementation, in which the analysis results are interpreted. Relying on the previous steps, the focus is on the properties of the architecture, in the context of product architecture management. Those are represented mainly by the structural characteristics of the architecture, and secondly on the properties resulting from the sum of entities of the architecture.

The goal of the architecture analysis step can be described as the grasping of system properties, based on the provided system representation of the MDM model. In detail, the overall goal can be described as the identification of intra- and inter-domain characteristics of the architecture, on one hand, and the identification of boundary conditions and degrees of freedom on the other. The characteristics of the architecture are represented by properties of entities and overall properties, as well as structural characteristics identified within and across domains. Boundary conditions and degrees of freedom as a result of analysis enable the synthesis process to concentrate on possible solutions, while at the same time, the underlying restrictions of boundary conditions become transparent.

The most relevant entities in this step are the system of goals (what is to be analyzed) and the system of objects (where within the system can the solution be found). The system of action (how the solution can be realized and which boundary conditions exist) supports the process. In general, as with the previous steps, the focus is on the relevant domains identified within the goals and objectives step.

Step 5: **Architecture Analysis**	
Goals	• Identification of intra- and inter-domain characteristics of the architecture • Identification of boundary conditions and degrees of freedom
Architecture Entities	• Goals and object system as priority • Action system • Focus on domains relevant for project objectives
Methods	• Structural analysis by means of Graph Theory and matrix methodology (techniques, characteristics, and metrics)
Models	• Matrix and graph representation • Analytical models
Critical Aspects	• Analyzing reasonable level of abstraction • Consideration of inter-domain relations • Focusing onto relevant domains with respect to the objectives of analysis

Figure 7-11 Procedural model for the management of product architectures: Step 5 "Architecture analysis" – overview

The methods to systematically analyze a system were extensively introduced and discussed in chapter 5.4.3. The methods are differentiated into

- Techniques, i.e. analysis measures including rearrangements and visualizations of the system (such as clustering, banding, Δ- and ∑-MDM etc.),
- Characteristics, i.e. patterns within the structure, usually involving a number of entities and interrelations (clusters, leaf nodes, feedback loops, etc.)
- Metrics, i.e. numerical properties based on structural analysis that characterie the overall system or compare entities within the system (for example, the number of cycles, number of bridge nodes etc.)

The methods partly overlap; cluster analysis, for example, processes and rearranges the whole system, is also applied to characterize patterns within the system, and enables the comparison of elements, e.g. depending on the number of clusters they are in. The implications of the method will be discussed when introducing the application example in chapter 8.

The underlying models of the system are analytical, represented in graphs or matrices for more intuitive user perception. Due to the combination of the three models, the untrained user is capable of grasping and modeling the system, yet algorithms can be applied, which are not necessarily mathematically understood by the user. As such, tools are required to provide the analysis measures, which are nowadays numerous in both science and business.

For the successful system analysis, a number of critical aspects need to be considered. As for the modeling of the system and the precedent steps, the right choice of a reasonable level of abstraction is required to achieve meaningful results. While most of the discussed analysis methods deliver results for each domain, the interrelations between domains have to be considered, also based on the application of domain mapping logics. Finally, not all domains need to be thoroughly analyzed, since some domains provide information for interrelations within other domains (the coupling of components delivers relationships between requirements, for example). As a consequence, the user must focus on relevant domains with respect to the objectives of analysis.

Validation

The validation step was introduced to evaluate analysis results before implementation, i.e. before conclusions, based on the results are reached. Since analysis results provide the foundation for synthesis, on one hand, and strongly rely on the input information, on the other, a critical review before implementation is required. Since a reasonable outcome and sufficient results can only be achieved through the provision of correct input information, the validation step should provide the means to evaluate the analysis results and enable the assessment of the quality of information.

Step 6: **Validation**	
Goals	• Check plausibility of analysis results • Compare input (situation and objectives) and output information (analysis results)
Architecture Entities	• Focus on system of goals • All entities considered in previous steps
Methods	• Validation methods (e.g. plausibility checks, negation, cause-and-effect analysis, fault-tree analysis, etc.) • Transfer to source models
Models	• On basis of MDM-model • Source models
Critical Aspects	• Synthesis to be conducted, based on analysis results • Reliable results required for following steps • Results are dependant on input quality

Figure 7-12 Procedural model for the management of product architectures: Step 6 “Validation” – overview

While the following step of implementation aims for the comprehensive derivation of consequences from the analysis results, the goal of the validation step is to check the plausibility of analysis results, and compare the results with the situation and objectives defined in the first step.

Thereby, the architecture entities within the system of goals guide the step, while all entities considered in the previous steps are subject to validation. As was proposed for the goal-definition of this step, the comparison of the system of goals to the other architecture entities poses the major challenge in this step.

Different methods come into question for the validation of analysis results, which were mainly discussed in chapters 5.3.3 and 5.3.4, based on the discussions on requirements management, validation and verification. It has to be considered that the sufficiency of analysis results cannot be mathematically determined, which is why verification methods are discounted and validation methods are rather suitable. Typically, for the validation of vague requirements or expectations, a plausibility check is conducted on the basis of the stakeholder concerns and whether they are reflected by the analysis results. The transfer of the findings to source models makes analysis results accessible to the stakeholders, since familiar models are applied. To clarify the concerns and intentions of stakeholders in detail, the methods of negation, cause-and-effect analysis, or fault tree analysis can be used. All are similarly conducted. Identifying stakeholder concerns or requirements as a starting point, the cause-and-effect chains through the architecture framework can be easily revealed. Negation and fault tree analysis support the process by focusing on the consequences of a non-fulfillment of the requirements, functions, etc. On that basis, cause-and-effect chains can be validated and checked for plausibility.

Since the proposed architecture model is based on the identification of entities and interrelations, the identification of cause-and-effect chains is one of the core capabilities of the model. The analysis results can be checked for traceability and significance. If analysis results lack reasonable interpretation or meaning from a stakeholder perspective, the previous steps such as system decomposition and information acquisition need to be critically reviewed, to determine whether entities, interrelations, domains, or levels of hierarchies were missed during the first application of the procedural model.

Critical aspects of the first step are important for the following steps, such as synthesis, which are to be conducted based on analysis results. Given the importance, reliable analysis results are required for a successful execution of the following steps. Analysis results are highly dependent on the input quality of information, which can be checked for plausibility and quality using aforementioned methods.

Implementation

After the project scope has been defined, the system decomposed, modeled, and the analysis results validated, the implementation step links analysis to synthesis. Analysis results show the demand for action, boundary conditions, and degrees of freedom for synthesis. In some cases, the implementation of analysis results can follow immediately after the validation, since the scope and impact of analysis – hence the original problem – can be ad hoc grasped and solved by the designer. In other cases, the analysis results require a structured synthesis process, which guides the designer and enables a comprehensive and satisfactory solution of the original project objective.

Step 7: **Implementation**	
Goals	• Identify impact and demand for action on basis of analysis results • Define strategies to implement analysis results • Prepare synthesis phase
Architecture Entities	• All identified entities • System of action
Methods	• Interpretation of structural analysis and transfer to object • Scenario-definition and decision-making
Models	• MDM model • Source models
Critical Aspects	• Capability to interpret and concretize structural analysis results is required

Figure 7-13 Procedural model for the management of product architectures: Step 7 "Validation" – overview

It is the primary goal of the implementation step to identify and specify the demand for action and the impact of the analysis results. The implications are twofold. First, strategies to implement the analysis results need to be defined. For the immediate and manageable demand for action, synthesis can be conducted conventionally, based on the experience of the designers. In the case of complex problems and challenges, a detailed and structured synthesis process is required. In particular, if an intensive change of the product architecture is needed (e.g. for the purpose of radical innovations), a guided, thoroughly planned synthesis process is inevitable.

For the preparation of the synthesis phase, all architecture entities are relevant. The system of goals provides the directions and quantitative or qualitative targets, while the system of action gives boundary conditions for synthesis. The system of objects, on the other hand, provides the sphere of activity for the definition of solutions.

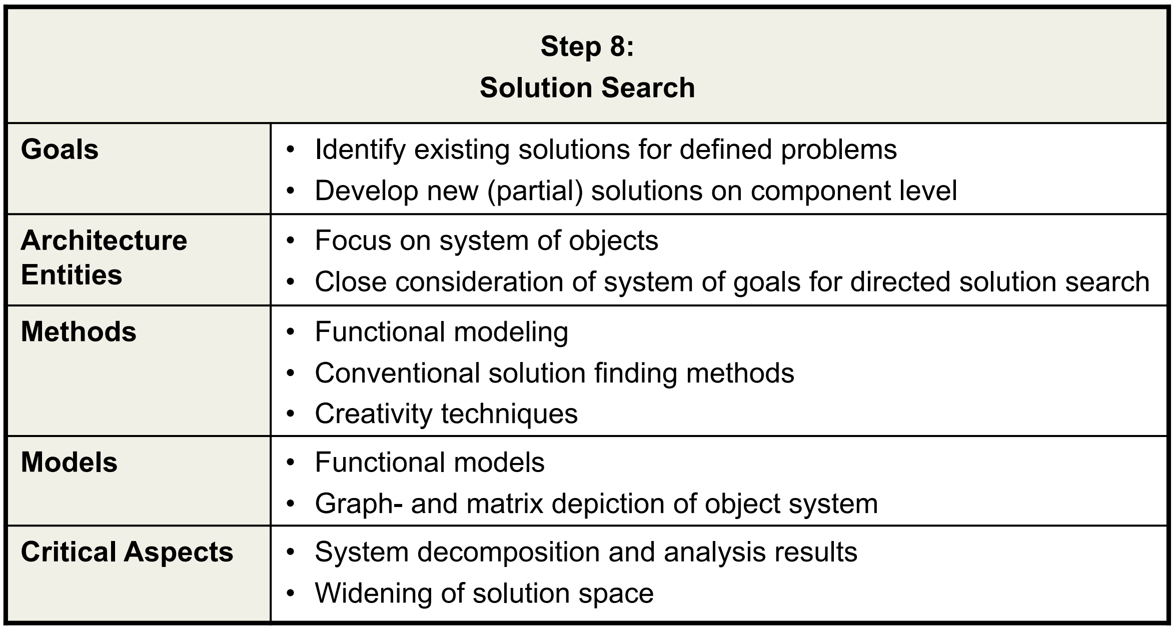
The suitable methods for the implementation of analysis results are largely dependent on the respective case. In any case, the interpretation of structural characteristics and other evaluation criteria (compare chapter 5.6.3) are required, independent of the scope of the required solution. Methods for decision-making, as discussed in chapter 5.6.2 come into question for product architecture management as characterized in chapter 5.6.1. Since synthesis is not conducted at this point, decisions are required based on results of analysis alone. If concrete scenarios emerge, based on the structural properties of the system, these can be analyzed using decision-making methods. Decision-making plays an important role throughout the process of design and development. At this stage, decisions are required regarding how synthesis will be conducted, on which levels and in which domains solutions it is necessary, which analysis results require structured synthesis, etc.

For the implementation and interpretation of analysis results as well as the decision-making, all types of models used to up this point can deliver valuable information. Both the source models and the MDM model help to clarify the situation's demand for action. While the MDM model contributes mainly structural information, more specific information and support for the interpretation of results can be derived from the source models, which usually serve a distinct purpose and are thus closer to the real system.

The implementation step is described as the transfer of interpreted analysis results to the system, to derive demand for action. The parallel discussion of structural and source models is essential. Equally critical for the implementation step is the capability to interpret and concretize structural analysis results, which is intended to be carried out through the application of systematic decision processes and the coupling of different models.

Solution search

The task of solution search, as the first step of synthesis, aims for the identification of solutions to the problems analyzed during analysis phase. For problems solvable on the solution level, the step of solution search might turn out to be sufficient; in other cases, solution search serves as a completion of the solution space for systematic synthesis.



Step 8: Solution Search	
Goals	• Identify existing solutions for defined problems • Develop new (partial) solutions on component level
Architecture Entities	• Focus on system of objects • Close consideration of system of goals for directed solution search
Methods	• Functional modeling • Conventional solution finding methods • Creativity techniques
Models	• Functional models • Graph- and matrix depiction of object system
Critical Aspects	• System decomposition and analysis results • Widening of solution space

Figure 7-14 Procedural model for the management of product architectures: Step 8 "Solution search" – overview

The overall goal of solution searching is the completion of the product architecture, based on available solutions inside and outside of the company. The identification of existing solutions is the first task, and is required to incorporate available entities into the product architecture model, thus completing the solution space. The design methodology supports the process of defining new solutions for the design task.

Thereby, the relevant architecture entities naturally center on the system of objects, where the entities contributing to the solution space are integrated. For the directed solution search, the system of goals provides guidance for synthesis, while the system of action supplies the boundary conditions for synthesis in general.

The methods for solution search were intensively discussed and evaluated in chapter 5.5. In particular, functional modeling (chapter 5.5.2) and conventional methods (chapter 5.5.3) come into question. Functional models thus serve as basis for solution searching. Decomposing the system from a functional perspective, building blocks on the detail-level can be identified, enabling the search for existing solutions, as well as novel ones. The search for existing solutions is conducted through conventional methods, based on sources of different kinds, both company internal and external. Creativity-supporting techniques (chapter 5.5.4) support the search for solutions by widening the scope and adding novel solutions to the solution space. Creativity-supporting techniques are capable of supporting the definition of novel solutions of limited complexity, and, as such, can contribute to solutions on the component rather than the architecture level. A precise problem description or problem decomposition is required a priori, which is provided by the exhaustive preceding analysis process.

Suitable models for the solution search step correspond with the proposed methods. Thus, functional models provide the basis for the solution search, while graph and matrix-depictions – the product architecture model – allow for a comprehensive overview and directed solution search within the product architecture model.

Critical for successful solution search is the thorough and precise system decomposition. Based on that decomposition, both functional and physical, analysis results are derived and point to the demand for action within the system. Finally, the outcome of the task of solution search, i.e. the widening of the solution space, is crucial for the following steps, the systematic exploration of the product architecture and the evaluation of solutions, to tap the full potential and provide a comprehensive and satisfying outcome of synthesis.

Systematic exploration

The systematic exploration follows the analysis and identification of solutions in the context of the solution search. After the objectives were clarified, the system decomposed, analyzed, and analysis results validated, the product architecture model was established and possible solutions for sub-problems identified or defined. Given these previous achievements, the solution space can be systematically explored to identify the most valuable solutions on the product architecture level, i.e. from a comprehensive view.

There are numerous goals of the systematic exploration of the solution space. Above all, the identified solutions must be placed into context within the product architecture model, both within their own domain, as well as in context of the other domains, such as requirements, functions, product family etc. Based on the comprehensive visualization, the impact and outcome of novel ideas in the different domains shows, for example possibilities to fulfill requirements, enlarging of functional capabilities, impacts on the product family etc. Based on the given requirements and functions, possible architectural solutions can be identified and completed, if additional solutions are required. Aside from the completion of architectural solutions, directed solution search can be conducted within all domains of the product architecture and the different levels of abstraction contained within it. For example, identified insufficiencies of the solution space can be solved on the basis of working principles, variation of functional possibilities, definition of new components etc.

Step 9: **Systematic Exploration**	
Goals	• Set identified solutions into context of all domains • Identify impact and outcome of novel ideas in different domains • Systematically complete possible architectural solutions • Search for solutions on all levels of the system
Architecture Entities	• Focus on system of objects • Close consideration of systems of goals and action for directed solution search
Methods	• Systematic synthesis methods • Comprehensive depiction of the solution space • Matrix- and graph-based methods • Computational synthesis
Models	• Graph and matrix depiction of solution space
Critical Aspects	• Completion (as far as possible) of solution space required • Rules and structural characteristics require valid input information

Figure 7-15 Procedural model for the management of product architectures: Step 9 "Systematic exploration" – overview

For a systematic exploration of the solution space, the system of objects has to be considered to provide the main relevant architecture entities. Solutions show in these domains, and a variation of the product architecture will be conducted within the system of objects. For the previous steps, the systems of goals and action provide boundary conditions and directions for directed and systematic solution search and need to be considered as well.

Two available methods for the systematic exploration of solutions are the introduced systematic synthesis methods, as discussed in chapter 5.5.5, such as different structuring schemes (e.g. morphological chart, classification tree, concept combination table, etc.), and matrix- and graph-based approaches, introduced in chapter 5.5.6 and 6.4. As the different discussed methods show, approaches are available for the different domains of the system of objects, such as working principles and effects, components, etc. Computational synthesis can be applied based on the defined model and analysis, and are able to define numerous solutions based on a defined rule-set (compare chapter 5.5.7). In the context of this work, computational synthesis is not the primary goal of application, yet can be conducted based on information inherited from the product architecture model. To cope with the methods and apply them in the most purposeful and directed manner, a comprehensive depiction of the solution space is necessary (see chapter 6.5). Therefore, available information, encompassing the whole product family within one domain (e.g. components, physical effects, functions etc.) can be visualized by the proposed product architecture model. For example, all components including their interrelations based on functional classification can be displayed, an overview on physical effects can be given, based on their input- and output-values, etc.

Accordingly, the most relevant model chosen for the step of systematic exploration is the graph and matrix depiction of the solution space, provided by the product architecture model.

On the other hand, it is critical for the systematic exploration of the solution space to have a rather complete availability – at least where required – of the solution space within the different domains and on different levels of abstraction. As for all models and methods, the choice of information, as well as the sufficient amount of information, is crucial,[93] yet can be supported by the visualization of information in the product architecture model. For the definition of rules and use of structural characteristics, the input-information is just as crucial. If automated procedures are to be applied, the definition of rules should be based on trustworthy input information, especially if the following decision-making is largely based on automatically generated results.

Evaluation and decision

Concluding the procedural model for product architecture management, the task evaluation and decision provide measures to rate the results of synthesis, which in turn were generated on the basis of analysis results and information within the product architecture model. As an outcome, the most promising solutions are identified, concretized and validated in the following detail design processes.

The goal of the evaluation and decision task is the preparation of the decision-making process, i.e. the collection of couplings between requirements and proper characteristics of the product architecture, which enable the validation of the fulfillment of requirements. This procedure aims for the qualification of informed decisions concerning the choice of product architecture solutions.

Based on the previous steps, the most relevant entities of the product architecture are those considered in the previous steps. As such, all entities of the product architecture come into question for the evaluation of solutions. The primary areas of focus is therefore on the coupling of requirements (stemming both from the system of goals and the system of action), on the one hand, and the physical product architecture, on the other. The preparation of informed decision-making thus includes the purposeful coupling of requirements to characteristic structures within the system of objects, such as functions, components, etc.

[93] Discussed for example in the context of modeling (chapter 4.2) and the principles of abstraction in chapter 5.4.2.

Methods for this step are summarized as methods for decision-making in chapter 5.6.2. While the principles of qualitative reasoning apply to the problem of product architecture management, the methods for single selection are suitable for smaller numbers of solutions. In contrast, the causality methods discussed in the respective section are comprehensive and applicable for the case of product architecture management. As the whole product architecture model, inheriting the entities and interrelations, is based on the idea of causal dependencies, the causality methods for decision-making can be easily connected to the product architecture model. For the definition of the evaluation criteria, the structural characteristics provide a reasonable means for evaluating product architecture,s based on the inherited structural properties of the synthesis results. Chapter 5.6.3 gives examples for the definition and use of structural characteristics as evaluation criteria.

Step 10: **Evaluation and Decision**	
Goals	• Prepare decision.making process • Enable informed decisions
Architecture Entities	• All considered entities • Coupling of goals and architecture characteristics
Methods	• Methods for decision making • Focus on structural characteristics
Models	• On basis of MDM model
Critical Aspects	• Availability and suitability of evaluation and decision criteria • Choice and prioritization of criteria • Quality of values

Figure 7-16 Procedural model for the management of product architectures:
Step 10 "Evaluation and Decision" – overview

Since causal chains based on numerous entities and their interrelations are the dominating scope of this step, the product architecture model based on the MDM modeling approach is the most suitable and efficient model to establish according evaluation criteria.

The most critical aspects for the successful evaluation of product architectures are the availability and suitability of evaluation and decision criteria. However, not all requirements can be evaluated based on the product architecture before detail solutions are available. In practical projects, the choice of solutions will be executed iteratively, focusing on the architecture-relevant criteria in the early phases, while reaching decisions in later phases, based on partial solutions on more detailed level. On the other hand, requirements might eliminate detail solutions in the early phase, e.g. based on working principles, which are out of bounds of the solution space. As for all evaluation or decision processes, the choice and prioritization of criteria are crucial to reach the desired results. Since all methods have to cope with this problem, a completely satisfying solution for this challenge is hard to define.

However, the comprehensive architecture model and the defined characteristics and criteria allow for the identification of interrelated criteria, thus supporting the prioritization of criteria. As a final critical aspect of the evaluation, the quality of values describing the characteristics has to be named. Clearly, the often-numerical outcome of analysis, and, as such, the values of evaluation criteria, strongly rely on the quality of input data. The too strong trust in quantitative results is seldom purposeful and has to be reviewed critically, especially in the early phases of product development. Combinations of criteria and comprehensive overviews of product architecture solutions are just as valuable as the prioritization and focus on central and highly relevant requirements.

7.5 Conclusion

The previous chapters introduced the comprehensive approach for the management of product architectures, based on three pillars. A framework for the coping with product architectures was introduced, providing the basis for the grasping and structuring of architecture entities and their interrelations. To cope with the framework, a modeling method for the product architecture was introduced, capable of capturing relevant product architecture information resulting from existing models and modeling techniques, and fulfilling the requirements of models in the context of systems architecting. Finally, the procedural model proposed relevant tasks and their interdependencies, interrelating the analysis- and synthesis processes. The iterative and recursive capabilities of the approach show in the combination of the three constituents. The comprehensive and consistent architecture model therefore provides the basis, coupling the domains and product architecture entities with one another. The different steps of the procedural model, which is not to be misunderstood as a strictly sequential process, fall back on these entities. As a result, iterative and recursive procedures can be conducted on the basis of the comprehensive and consistent architecture model.

The **product architecture framework** provides the basis for the project activities. Based on the literature reviews (chapters 1, 3, and 5), as well as project reviews during the research for this work, a comprehensive framework was established, allowing for a clear identification of relevant architecture entities and potential interrelations, and setting the boundary conditions for the project. Due to its generic character, the application can be defined according to the project, for example including sales, services, after sales or production and transport issues in the system of action and thus in the system of goals. The interrelations can be modified as well, yet a critical mass was defined and established on the basis of aforementioned reviews.

The **product architecture model** is a result of the identification of requirements in systems engineering in general (chapter 1.2) and in the modeling in the context of systems architecting (chapter 4.2), as well as the discussion leading to the established product architecture framework. The modeling approach chosen is capable of translating and interrelating existing models to provide a comprehensive and consistent product architecture model (see chapters 6.1, 6.2, and 6.3). The dual visualization through graphs and matrices provides intuitive interpretations for stakeholders from different disciplines, and the accessibility to elaborate mathematical analysis algorithms. Due to those properties of the model, the demand for continuity and the support of iterative and recursive procedures can be met.

The **procedural model** provides guidance to systematically cope with the framework and modeling approach. Therein, the application of different methods (compare chapter 5) is enabled and allocated to ten distinctive tasks in product architecture management. Whether all steps are required, and in which sequence, is determined by the respective project. This flexibility is realized by the comprehensive and consistent modeling approach. Again, it is important to recognize that the procedural model is not intended to be conducted sequentially without iterative and recursive application.

The following chapter will validate the approach for product architecture management on the basis of a project in the automotive industry. All steps will be addressed and conducted there; however, not all aspects can be covered. This again documents the flexibility of the approach, depending on the case of application.

8. Validation

The previous chapters elaborated methods and approaches for the management of product architectures, which already exist in different disciplines. The product architecture framework, model, and procedural model were introduced to tie together existing approaches and models, and establish a comprehensive framework for the numerous interrelated phases of product architecture management. To complete the approach, and in follow up to the previous prescriptive study in chapters 6 and 7, the descriptive study in this chapter provides a validation example of the defined approach. The presented case study alone cannot cover all aspects of the approach; it does, however, point out the focal ideas and the general practicability and validity of the approach. It should be considered as one of many use case-based studies conducted during and after the definition of the overall approach for the management of product architectures.

8.1 Case study: automotive drivetrain development

As an introduction to the validation example, the following sections will roughly outline the situation and problem-description, before the following chapters start use a hands-on approach with the procedural model to apply the framework and model to the problem. The example of the automotive drivetrain was chosen, since the current political and social situation demands radical innovations in that sector, yet the potential of current technologies is almost fully tapped. As a result, variations of the product architecture are required to provide radical innovations, based on existing mature technologies.

The increasing shortage of fossil energy resources, the increasing fuel costs as a result, stricter emission legislation, and increasing demands for safety, all call for highest efficiency in the context of energy consumption; these changing conditions impact the situation of the transport sector in general, and the automotive sector in particular. Since customers in the automotive sector demand high performance at the same time, the automotive industry has to deal with this conflict of targets and provide sustainable long-term solutions for future customers [LIEBL 2006].

Motivated by this situation, studies were conducted with a number of different targets. The focus of all considerations, and as such the boundary condition and system border, is the automotive drivetrain, i.e. the tank-to-wheel flow of energy and related requirements, physical entities, functions, and properties. Other areas contributing both to the solution of the conflict of targets and to efficient dynamics in general were not considered intensively within the scope of the following considerations. Examples of these side issues are aerodynamics or lightweight design.

The different targets will be clarified as part of the first step of the procedure, while the overall goal is made clear in advance: the overall goal of the study is the analysis of existing drivetrain configurations, their physical entities and entities under consideation. The function-domain of drivetrain concepts is to be analyzed similarly. Based on the analysis results, structured synthesis within all entity domains is to be conducted, i.e. on functional and physical level(s). Aiming for the definition of coherent drivetrain concepts, solutions are to be evaluated by functional capabilities and further properties. Since the evaluation through simulation is time-intensive and costly, the focus is placed on the early phases of development to provide rudimentary directions of possible solutions, based on key figures.

8.1.1 Goals and objectives

As proposed in the procedural model of the product architecture management approach, it is the goal of the first step of the procedure “Goals and objectives” to identify project goals, situation, stakeholders, use cases, relevant domains, and critical cause-and-effect chains. Since the objectives for the presented case study are on project level, rather than product level, typical requirements management methods are applied marginally, while the focus is placed on use cases and scenarios. The results are documented accordingly, depicting outlines of use cases and scenarios. As a starting point for the MDM model, the meta-model for the overall system is defined.

With the overall goal defined in the project description, a number of subordinate goals can be derived, which are summed up in the following points:

- Analysis of existing drivetrain concepts on the basis of their functional and physical properties
- Decomposition of the functional and physical structure down to elementary building blocks
- Identification of key properties of building blocks
- Analysis of existing concepts and derivation of evaluation criteria
- Validation of evaluation criteria, based on simulation and research results
- Analysis of how far waste-energy can be recovered through the introduction of additional systems
- Search for alternative solutions of drivetrain concepts on all levels under consideration
- Evaluation and comparison of defined solutions and decisions, based on scenarios and use cases
- Identification of possible modularity scenarios based on analysis and product family restrictions

In the early phase, possible geometrical and commercial aspects were ignored, focusing solely on the functional potential of solutions. As such, the fulfillment of functional requirements and their combination was the focus of consideration, as well as the identification of potential in terms of degrees of efficiency, boundary conditions etc.

Since the complex subject of the automotive drivetrain and its evaluation can, in the end, only be tackled through comprehensive simulation and design measures, the results presented here can only provide the boundary conditions and point out promising directions. The different use cases in particular, i.e. the usage of the automobile in different situations and the underlying drive cycles, provide boundary conditions, which are inevitable for a comprehensive analysis and evaluation. To grasp this complexity in the early phase, a scenario of the automotive environment was established to identify key factors characterizing the use of the automobile in the future. The scenario includes about 80 factors, grouped by the classes depicted in the following table.

Table 8-1: Classes of influence factors on the automotive environment

Resources and Technologies
Availability of energy and resources
Usage of energy and resources
Technologies
Sociodemography
Population
Migration balance
Population density
Settlement pattern
Number of households
Educational level
Age pattern
Life expectancy
Legislation
Emission
Vehicle
Pollution control
Taxes
Road traffic regulations

Transport Structure
Mobility
Infrastructure
Market and competition
Characteristics of the automotive sector
Market situation and terms
Economy
Gross domestic product
Private consumption
Consumption structure
Governmental consumption
Import and export
Price level
Wages
Labor productivity
Labor market
Net household income

To identify critical factors for the definition of coherent scenarios, a structural analysis of the network of factors was conducted. The network of factors was defined based on literature reviews, workshops, and interviews. These factors are then interlinked by directed dependencies of the type “influences”. A number of analysis criteria come into question. Leaf-nodes, for example, characterize active or passive factors, depending on the direction of the dependencies. The active- and passive-sum of all nodes describes their general behavior within the network. Typical scenario techniques provide grids for the evaluation of criteria, both direct and indirect, to systematically identify the key factors [compare GAUSEMEIER et al. 1996, MIßLER-BEHR 1993]. The network of influence factors provides the basis for these.

The following figures depict the underlying data schematically, showing the impulsive factors to be the most relevant. Impulsive factors show in all categories; highly relevant and of immediate influence are those of the transport structure, which are also highest in number. The following selection of factors was identified as relevant for the definition of realistic drive cycles: number of vehicles, traffic volume, density of traffic, length of traffic routes, typical path length, traffic management, and traffic performance.

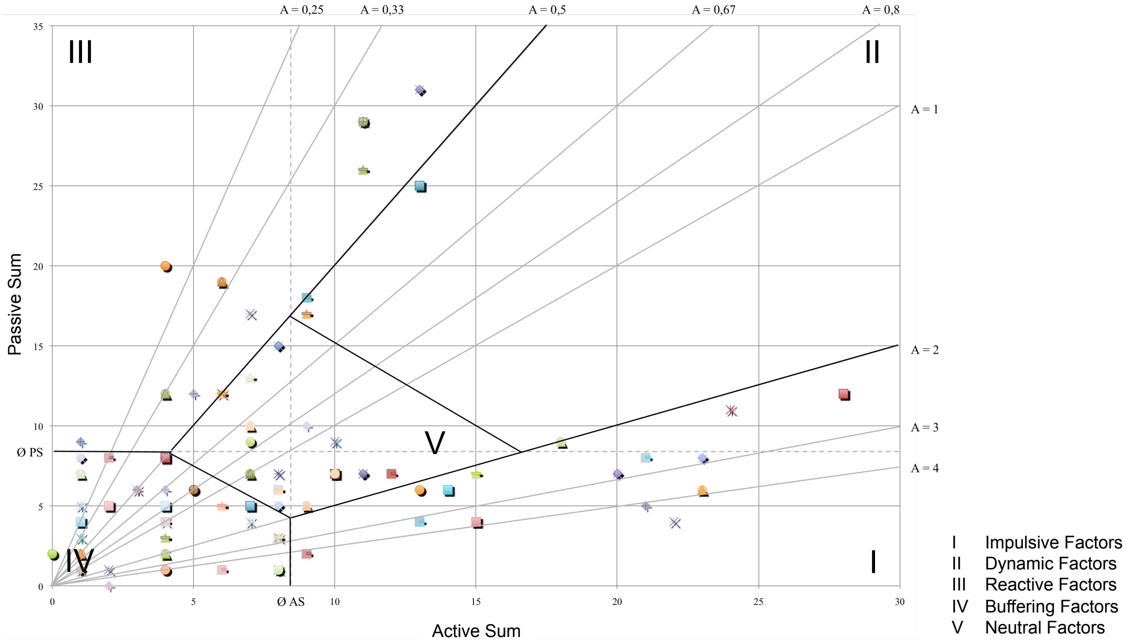


Figure 8-1 Grid of influence factors

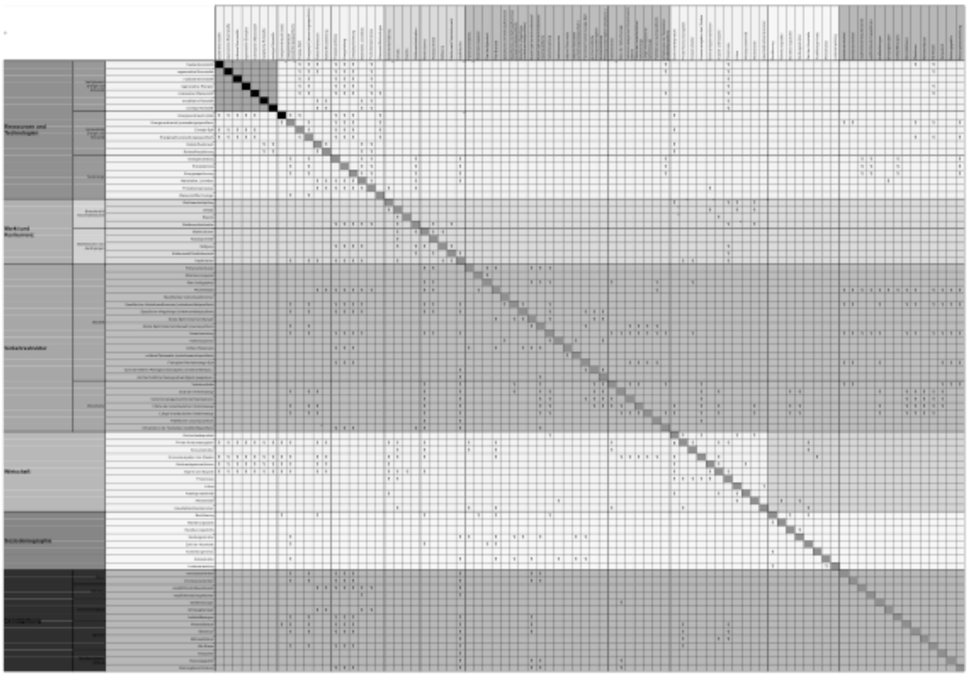


Figure 8-2 Matrix of environmental influence factors for scenario definition

The chosen scenario is characterized as a likely scenario in the near future, with only moderate changes to the key factors. Trends indicate that the number of vehicles, traffic volume, density of traffic, traffic management measures, and traffic performance will rise moderately, while the effects of urbanization, i.e. many short routes in cities and long routes from rural zones to the cities, characterize the length of routes and paths. Since the underlying drive cycles are largely relevant, use case-based scenarios were defined. The precise drive cycles are only of limited use in the early phase, but allow for the proportional differentiation of driving modes (brake, accelerate, etc.). The necessary information is therefore derived from known drive cycles and simulations based on standardized drive cycles.

In accordance with the described project objectives, the relevant domains and their interrelations were identified; this is depicted in the following meta-model of the presented case study. The meta-model is based on the defined architecture framework, detailing the domains and their interrelations based on the case study. The main cause-and-effect chain in the following section describes the main relevant interrelations.

The identified use cases define the resulting customer requirements in the context of that use case. The customer requirements, in return, cause performance and functional requirements. Those requirements demand for either certain properties or functions to be fulfilled. Within the domain of functions, primary functions require secondary functions, while both may cause harmful functions. Additionally, secondary functions may be introduced to prevent harmful functions. To fulfill the demanded functionality and to define properties, the domains of physical entities – components and assemblies – are introduced, also potentially causing harmful functions. Within the domains of physical entities, components result in assemblies. Physical effects are used by the physical entities and enable the functionality. Properties resulting from the physical entities and functions can then be validated against the requirements regarding their fulfillment. The product family and use cases both require and use the physical entities.

		RC	RP	RF	CC	CA	W	FP	FS	FH	PP	PC	LU	LF
RC	Customer	▨	"cause"											
RP	Performance		▨								"demand"			
RF	Function			▨				"demand"						
CC	Components				▨	"result in"	"use"	"fulfill"		"cause"	"define"			
CA	Assemblies					▨								
W	Effect						▨	"enable"						
FP	Primary			"fulfill"				▨	"require"	"cause"				
FS	Secondary								▨	"cause" "prevent"				
FH	Harmful									▨				
PP	Properties	"are validated against"			"result from"						▨			
PC	Characteristics											▨		
LU	Use Cases	"define"			"require"								▨	
LF	Product Family				"use"									▨

Figure 8-3 Meta-model for the presented case study

On the basis of the introduced meta-model and objectives, the following steps can be conducted, starting with the system decomposition, i.e. the identification of entities of the system and their interrelations.

8.1.2 System decomposition

For system decomposition, the levels of abstraction within domains (if not already defined in the meta-model) have to be identified, and the existing entities classified and grouped within them. According to the objectives of the case study, the hierarchies of physical entities and functions are more elaborate than those of the other domains.

The **requirements** domains alone represent only a small number of entities. The leading customer concern is the energy-efficient transportation in different use cases, i.e. situations or drive cycles, and a dynamic performance at the same time. As a third requirement, the cruising range of the vehicle has to be taken into account. The customer requirements directly cause performance and functional requirements. Performance requirements include acceleration, top speed, and energy efficiency. As functional requirements, the demand for different driving modes results from the customer requirements.

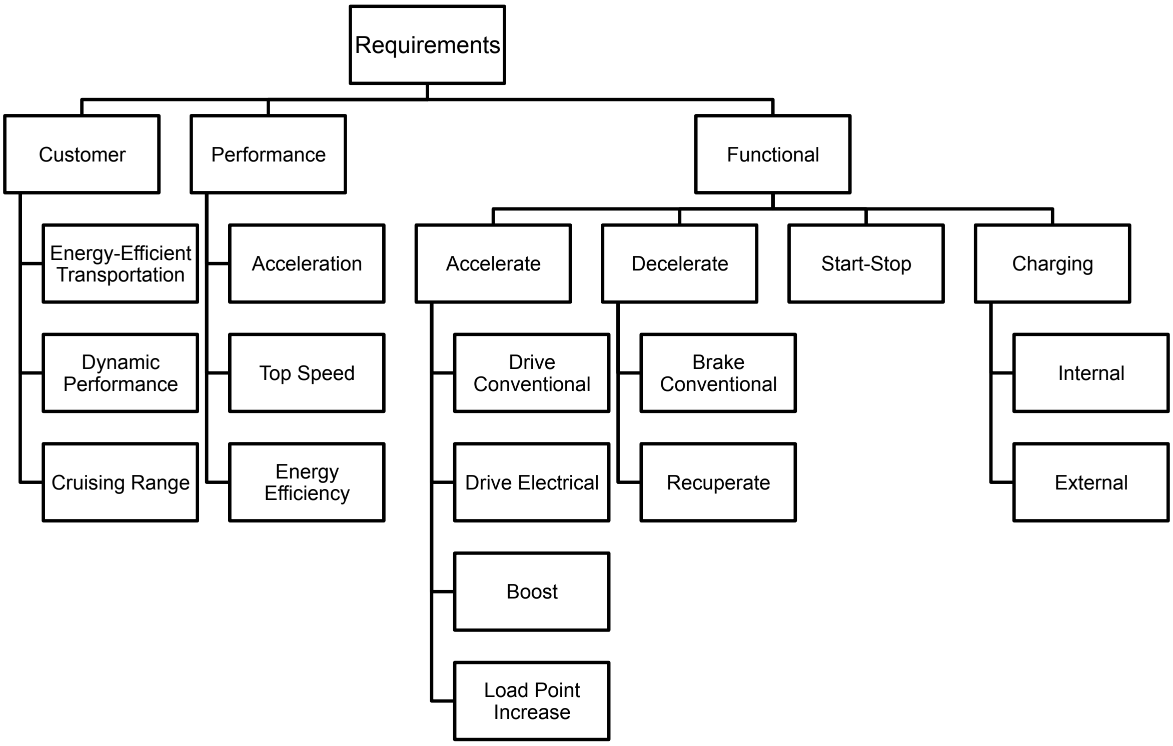


Figure 8-4 Decomposition of Requirements

The relevance of the requirements, especially functional requirements, largely depends on the use case, i.e. functional requirements are weighted differently for diverse requirements. The performance- and customer-requirements largely reflect in the properties of the architecture.

The domains of **physical entities** are largely based on a classification of entities on the component level. A differentiation is made between energy storage and energy conversion components. While the following figure depicts the different classes of domains, the single entities are not depicted. Entities, i.e. existing components are assigned according to their main purpose. A combustion engine for example converts chemical to mechanical energy, but also to thermal energy as a side effect.

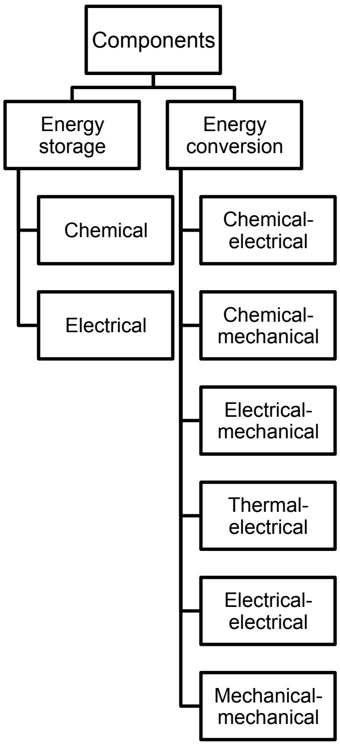


Figure 8-5 Decomposition of Components

Assemblies are intended to be a result of the analysis of the architecture and thus are not defined from the start. In general, assemblies are considered to be a grouping of components.

The domain of physical effects is not further decomposed. Physical effects in general are characterized by their input and output parameters, the respective amplitude, and the underlying physical formula [compare PONN & LINDEMANN 2008, pp. 311 ff.].

The **functional** system and hierarchy was established on three layers, all below the main function "move vehicle". The primary functions are "store", "convert", and "use/transmit" energy. Primary functions can be further detailed, based on the specification of the functions, as depicted in the following figure. Secondary functions, for example "dissipate waste heat", follow the same classification as the primary functions. Harmful functions are not further decomposed at this point; a decomposition of harmful functions will be conducted on the basis of the information acquisition.

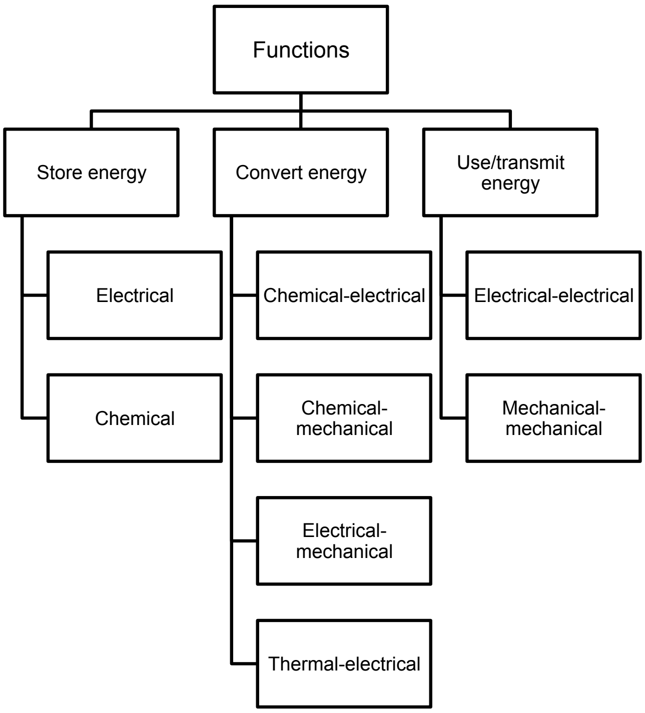


Figure 8-6 Decomposition of Functions

Use cases and the product family are not primary subjects of consideration for the case study, and are thus not further decomposed, but decomposed when required in the following steps.

The system decomposition reflects the early perception of the system. As was discussed during the introduction of the procedural model, iterative and recursive procedures are common, especially among the first three steps.

8.1.3 Information acquisition

The information acquisition aims for the identification of entities and the interrelations between entities for the previously decomposed domains. The upcoming sections will provide examples regarding how information for the case study can be acquired; the inter- and intra-domain interdependencies are of special interest.

For the systematic identification of functions of existing drivetrain concepts, different functional models were established. The flow-oriented functional model provides the material- and energy-flows of the system. The flow-oriented model differentiates between operations and states combined with functions (see chapter 5.5.2), while for further processing, the operations are considered as functions. The figure on the following page depicts an example of the flow-oriented functional model of a parallel hybrid, combining the conventional drivetrain (top of the figure) with the electrical drivetrain, which in the given example is also externally chargeable, as a plug-in hybrid.

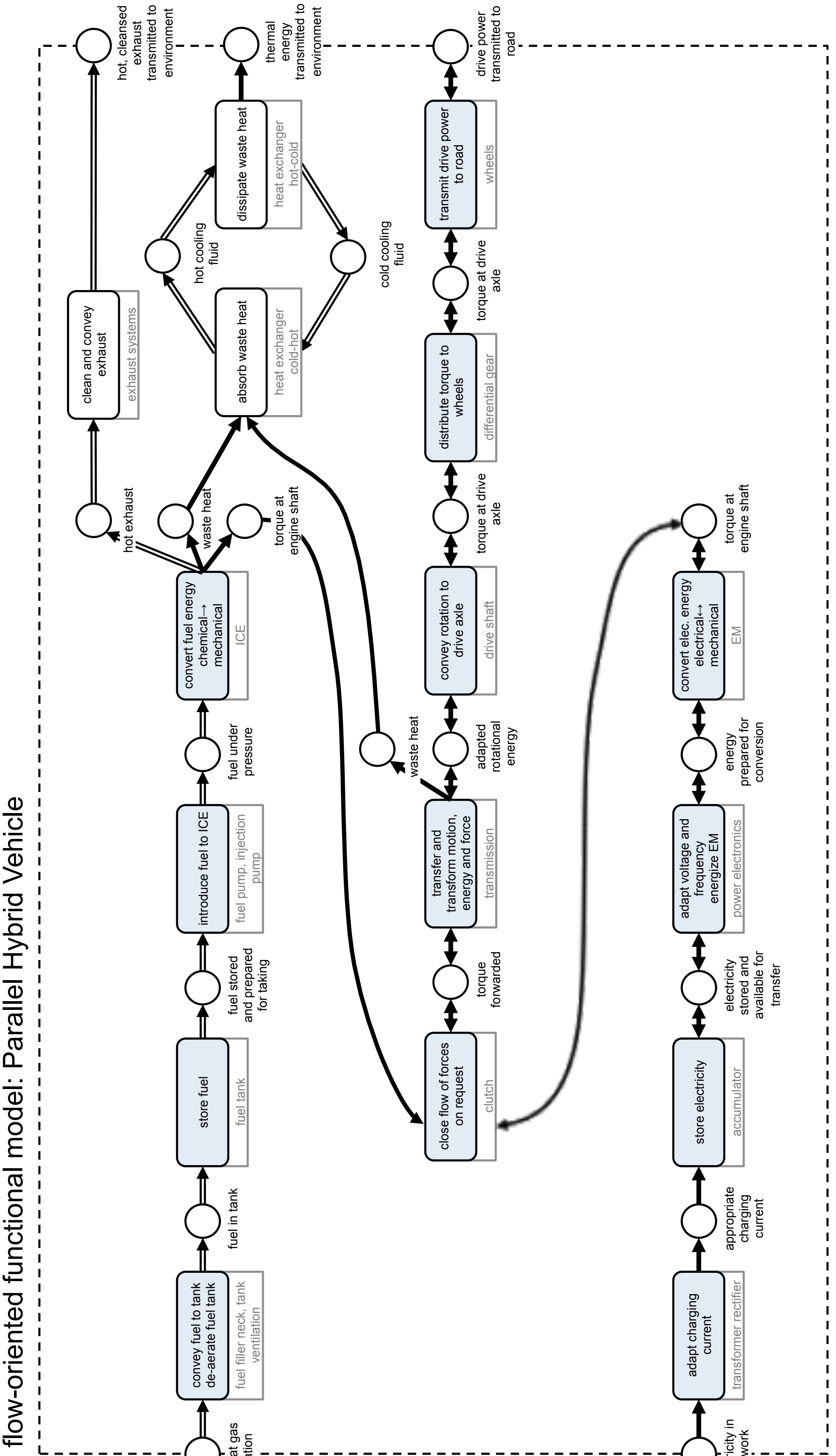

flow-oriented functional model: Parallel Hybrid Vehicle
fuel at gas station
convey fuel to tank de-aerate fuel tank
fuel filler neck, tank ventilation
fuel in tank
store fuel
fuel tank
fuel stored and prepared for taking
introduce fuel to ICE
fuel pump, injection pump
fuel under pressure
convert fuel energy chemical→ mechanical
ICE
hot exhaust
waste heat
torque at engine shaft
clean and convey exhaust
exhaust systems
hot, cleansed exhaust transmitted to environment
absorb waste heat
heat exchanger cold-hot
hot cooling fluid
dissipate waste heat
heat exchanger hot-cold
cold cooling fluid
thermal energy transmitted to environment
close flow of forces on request
clutch
torque forwarded
transfer and transform motion, energy and force
transmission
waste heat
adapted rotational energy
convey rotation to drive axle
drive shaft
torque at drive axle
distribute torque to wheels
differential gear
torque at drive axle
transmit drive power to road
wheels
drive power transmitted to road
electricity in network
adapt charging current
transformer rectifier
appropriate charging current
store electricity
accumulator
electricity stored and available for transfer
adapt voltage and frequency energize EM
power electronics
energy prepared for conversion
convert elec. energy electrical↔ mechanical
EM
torque at engine shaft

Figure 8-7 Flow-oriented functional (example of parallel hybrid)

Single lines within the model represent energy-flows, while double lines represent material flows (i.e. fuel or air). The chosen level of abstraction largely influences the nature of a functional model. An engineer involved in drivetrain development might find the model far too abstract. For the presented case study, which is aiming for an understanding of existing architectures and the systematic synthesis of architectures, the level of abstraction appears to be sufficient and reasonable. The model can naturally be detailed at will. For example, the functional model for a clutch alone can be depicted in more detail than presented example of the overall drivetrain.

The allocation of components to the functional model (in the given example implemented below each operation) is not uncommon. Clearly, not every system allows for the one-to-one allocation of components to functions. In the given example, components may be either allocated repeatedly to more than one function, or more than one component assigned to one function.

Summing up, the functional model provides reasonable input for the interrelations between functions, as well as the interrelation between functions and components. A solid basis is established for the analysis and comparison between different drivetrain concepts. To achieve a reasonable level of detail when modeling nevertheless requires experience and might require reworking and iterations.

In the figure on the following page, the same system is depicted in the form of the relational functional model (see chapter 5.5.2), introducing the harmful functions. The information of the flow-oriented functional model was used as a starting point for the relational functional model. The functions of the flow-oriented functional model are found within the relational functional model, in which additional (secondary) functions appear, as well as harmful functions. As a result, both types of functional models are interrelated and provide information for the architecture model. Harmful functions in particular point out the downsides of existing architectures and components. Additionally, directed solution search, as proposed in the TRIZ methodology, is prepared. The coupling of components to the functional model can also be conducted.

The information acquisition for components can be conducted in three ways. The allocation to the functional model was already conducted. The characterization of components and the analysis of existing architectures will be shown in the following section. For the case study, the numerous types of components identified during system decomposition are characterized, to enable capturing the properties of the system based on single components. The following table shows examples of the property categories for electrical energy storages, i.e. batteries. The values for the properties can be identified for the class by providing a value range, or for an individual component with defined effects.

Table 8-2: Properties for component classes (example of electrical energy storages)

Electrical energy storages
Specific power (W/kg)
Volumic energy (Wh/kg)
Charge efficiency (%)
Energy efficiency (%)
Open circuit voltage (V)
Operating temperature (°C)
General properties

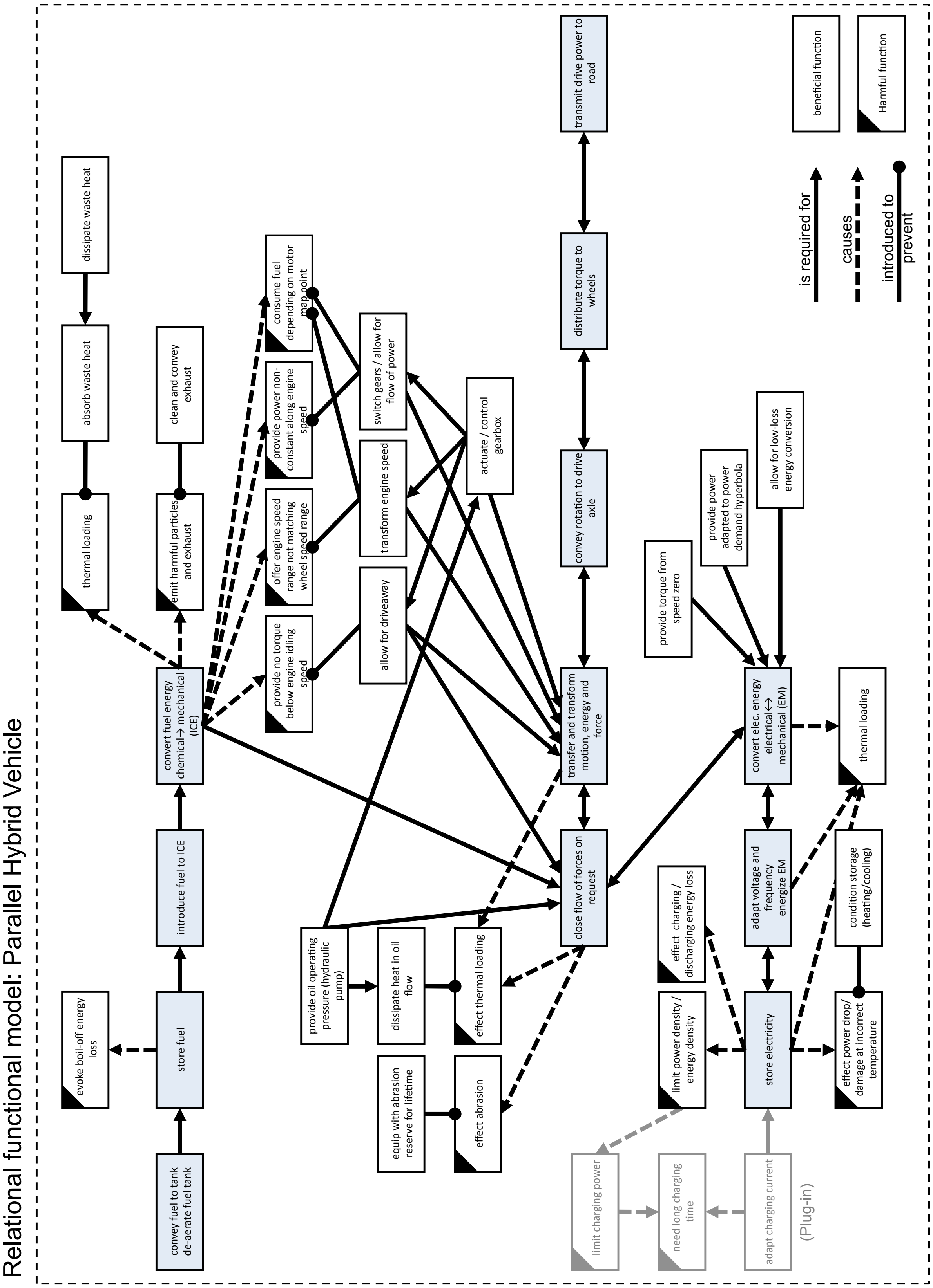

Relational functional model: Parallel Hybrid Vehicle
convey fuel to tank de-aerate fuel tank
store fuel
evoke boil-off energy loss
introduce fuel to ICE
convert fuel energy chemical→ mechanical (ICE)
thermal loading
absorb waste heat
dissipate waste heat
emit harmful particles and exhaust
clean and convey exhaust
provide no torque below engine idling speed
offer engine speed range not matching wheel speed range
provide power non-constant along engine speed
consume fuel depending on motor map point
allow for driveaway
transform engine speed
switch gears / allow for flow of power
actuate / control gearbox
provide oil operating pressure (hydraulic pump)
dissipate heat in oil flow
effect thermal loading
equip with abrasion reserve for lifetime
effect abrasion
close flow of forces on request
transfer and transform motion, energy and force
convey rotation to drive axle
distribute torque to wheels
transmit drive power to road
limit charging power
need long charging time
adapt charging current
(Plug-in)
limit power density / energy density
effect charging / discharging energy loss
store electricity
adapt voltage and frequency energize EM
convert elec. energy electrical↔ mechanical (EM)
provide torque from speed zero
provide power adapted to power demand hyperbola
allow for low-loss energy conversion
thermal loading
condition storage (heating/cooling)
effect power drop/ damage at incorrect temperature
is required for
causes
introduced to prevent
beneficial function
Harmful function

Figure 8-8 Relational functional model (example of parallel hybrid)

After existing architectures were assessed from a functional perspective and on the component level, the physical architecture of the drivetrain is considered for information acquisition. As an example, the following architecture diagram of a parallel hybrid completes the picture of the two functional views and the component view.

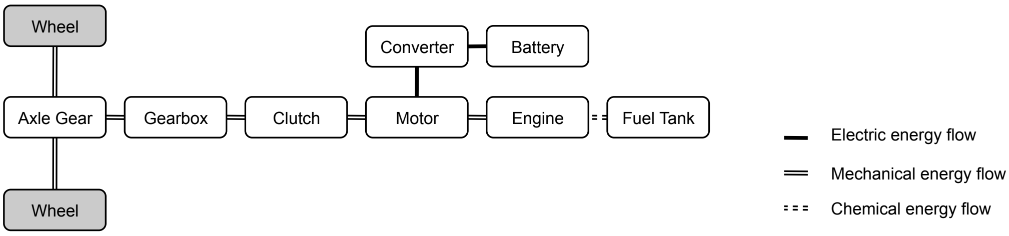


Figure 8-9 Architecture diagram (example of a parallel hybrid vehicle architecture)

Compared with the flow-oriented functional structure, the analysis of the presented physical architecture provides similar knowledge, since the focus is placed on the functional interaction between the physical system elements. According to the objectives, the focus was on the energetic-functional perspectives. Additional knowledge is likely to be built based on models from different viewpoints. For the synthesis phase, the previously discussed models provide a valuable and informative basis, as the following sections show.

The result of information acquisition is a comprehensive overview of the existing entities and entity-relationships, based on different viewpoints of the different levels of abstraction in the system. The presented functional models and architecture diagrams were established for known drivetrain concepts and the properties of physical entities researched, by means of examples from different makers. The information is modeled and analyzed in the following section, preparing a systematic synthesis and a structured overview of the potentials and restrictions for development.

8.1.4 System modeling

While the source models were used for the information acquisition process, the transfer to MDM notation is conducted as a next step. The underlying principle was introduced in chapters 6.1 and 6.2.

The modeling in MDM notation follows strict rules and requires that the information be prepared accordingly. The semantics of models, such as the functional models, regularly meet the MDM notation halfway, since types of entities and interrelations are precisely defined. However, the relation between models, if not given through the collective use of the same entities, such as functions, has to be established. In the given example, the models used for information acquisition do not provide the interrelations between the requirements and the functional and physical entities. Methods such as FMEA or QFD support the definition of these interrelations, yet demand a demanding procedure if the outcome of the methods is not explicitly required. To complete the picture defined by the meta-model, the lack of information can be directly filled in within the matrix-model. To be able to estimate which information in particular is required, the analysis must be thoroughly planned.

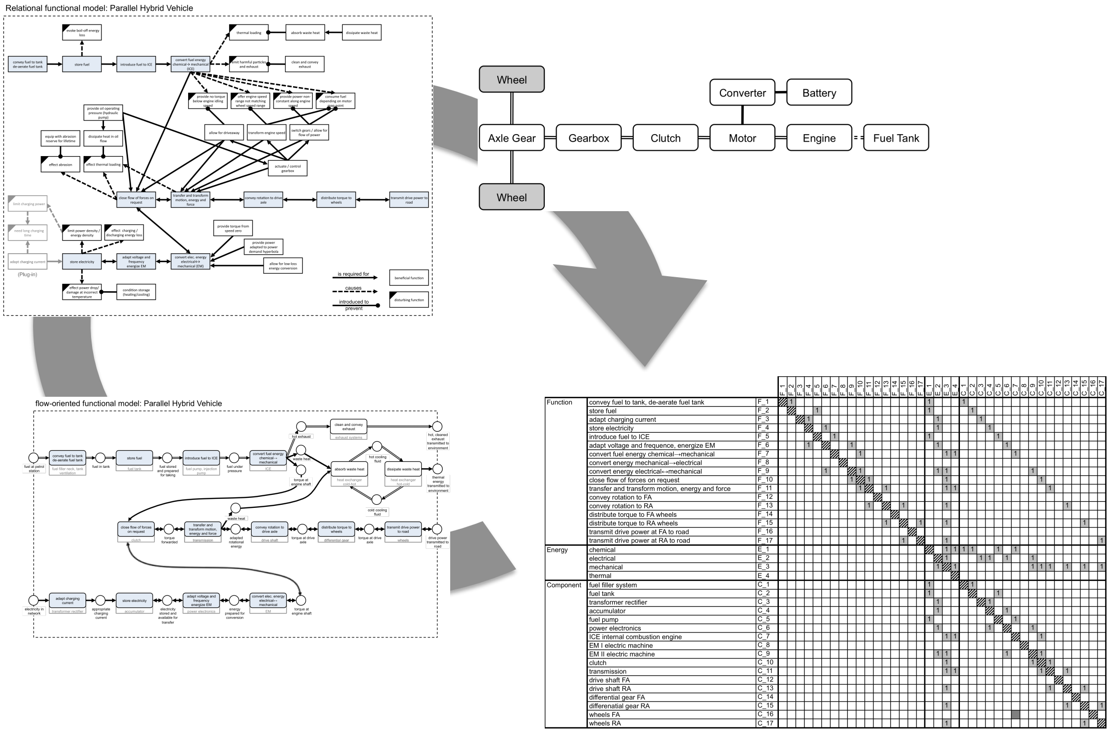

Relational functional model: Parallel Hybrid Vehicle
flow-oriented functional model: Parallel Hybrid Vehicle
Wheel
Axle Gear
Gearbox
Clutch
Motor
Engine
Fuel Tank
Converter
Battery
Wheel
Function
Energy
Component
convey fuel to tank, de-aerate fuel tank
store fuel
adapt charging current
store electricity
introduce fuel to ICE
adapt voltage and frequence, energize EM
convert fuel energy chemical→mechanical
convert energy mechanical→electrical
convert energy electrical→mechanical
close flow of forces on request
transfer and transform motion, energy and force
convey rotation to FA
convey rotation to RA
distribute torque to FA wheels
distribute torque to RA wheels
transmit drive power at FA to road
transmit drive power at RA to road
chemical
electrical
mechanical
thermal
fuel filler system
fuel tank
transformer rectifier
accumulator
fuel pump
power electronics
ICE internal combustion engine
EM I electric machine
EM II electric machine
clutch
transmission
drive shaft FA
drive shaft RA
differential gear FA
differential gear RA
wheels FA
wheels RA

Figure 8-10 System modeling – principle

The above figure shows the principle of transferring the models into the matrix notation. The MDM depicts functions, components, and the energy states as supplements for the functional and component descriptions. The matrices can be analyzed in the following steps. A potential summary of the individual systems in matrix notation results in a comprehensive model of the actual known solution space.

While the matrix in above figure displays the functional model, its energy states and components in an MDM model, other dependencies on the level of entities and interrelations cannot be as clearly grasped. For example, the functional requirements, i.e. the different driving modes of potential vehicle architectures, cannot be mapped onto individual entities – on neither the functional nor component level. Considering the system modeling as a planning phase of the architecture analysis, the solution to such challenges can be prepared. In the following section, a possibility is described using the example of functional requirements in the case study.

The functional requirements were grouped under the four classes of acceleration, deceleration, start-stop, and (electrical) charging. The entities reflect the usual driving modes considered in automotive drivetrain development, while the conventional braking and start-stop will be factored out for the following considerations, discussing the seven remaining functional requirements. The goal is to reflect the functional requirements within the functional domain, which is composed exclusively of energy flows. The solution lies within the previously introduced functional models. A distinct path within the functional system can depict functional requirements. The function of conventional driving, for example, can be depicted by a path from fuel tank to wheels. The dependencies after the internal combustion engine are required to be only mechanical. A transformation to electrical energy would result in the function of driving electric within a serial hybrid.

The following table displays the functional requirements, the structural characteristics within the functional system, and the restrictions for the successful identification of functional requirements being fulfilled within the system.

Table 8-3: Functional requirements and mapping to the functional domain

Functional requirements	Structural Characteristic within the functional domain	Restrictions
Drive conventional	Existing path from fuel tank to wheels	No transfer to electric energy
Drive electric	Existing path from battery to wheels	-
Boost	Existing paths “drive conventional” and “drive electric”	Paths must be technically overlayable
Load point increase	Existing paths “drive conventional” and “internal charging”	Both paths must found on the same engine node as the converter
Recuperate	Existing mechanical path from wheels to electric motor	-
Internal charging	Existing path from the chemical-mechanical converter to battery	Requires two electric motors to drive electric while charging
External charging	Direct interrelation between external source and battery (through converter)	-

As a result, the path analysis for referenced entities can deliver potential solutions, which again have to be checked for fulfillment of the given restrictions. Existing solutions, as well as solutions defined during synthesis phase, can be evaluated by means of the introduced structural criteria.

The above section gave an overview of how the system is modeled, and how interrelations between domains can be deduced, even if not displayable on sole interdependencies. The following section describes potential analyses of the architecture, based on the given information.

8.1.5 Architecture analysis

The architecture analysis in the presented case study can be conducted in two stages. First, existing solutions can be analyzed separately from one another, allowing for a comparison based on the properties which result from the functional and component domain. Second, the existing solutions can be summarized into one model using the ∑-MDM approach. Based on the summary, the shortcomings and restrictions of all existing solutions can be derived. To set the evaluation into context, the different use cases (announced in chapter 8.1.1) can be consulted.

Table 8-4: Analysis criteria for individual architectures

Analysis criterion	Domains	Relevant structural characteristics
Number of driving modes	Functions	Path
Quality of driving modes	Components and properties	Path and path-length attributes
Structural complexity index	Components	Number and variety of entities and interrelations
Modularity	Components and functions	Cluster analysis

The analysis of architectures allows for the identification of the **number of driving modes**, i.e. the covering of the functional requirements, as discussed at the end of the previous chapter. The **quality of driving modes** can be estimated based on the properties of components being part of the path. For example, comparable values between elements based on weights or costs can be derived. The potential of the degree of efficiency of the paths is of the greatest interest. It is clearly only the potential of the degree of efficiency that can be evaluated, since its actual values depend largely on the respective load point and other dynamic factors, which cannot be generalized. The architectures can thus be compared based on vague estimates relatively to one another, yet a conclusive decision can only be made on the basis of the absolute values of more sophisticated concepts.

Since available space and packaging is always an issue in automotive development, a **structural complexity index** was defined, based on the number and variety of entities and interrelations. The underlying hypothesis states that the greater the number and variety of the elements to be combined, the more difficult the realization becomes in terms of cost, effort, and space. However, the best solution may not be the one with the smallest number of elements, since an imaginative solution might incorporate more components into one (the two-mode gear box, for example), solving the dilemma underlying the hypothesis. As with the other criteria, they can only be used as a guideline for the identification and comparison of potential solutions.

Modularity, as a last criterion, aims for the evaluation of potential solutions against the existing product family or platform concept. The cluster-analysis shows potential interfaces and modules, and enables the drawing of different conclusions. First, if the architecture is compatible with the existing product family from a structural perspective, and second, if potential solutions can be realized together within one product line or product family.

Table 8-5: Analysis criteria for the sum of considered architectures

Structural analysis criterion	Domains	Relevant structural characteristics
Degrees of freedom	Components and functions	Number of alternative paths
Restrictions	Components and functions	Bridge nodes and edges, Cluster analysis, paths
Potentials	Harmful functions	Active- and passive sum
Modularity	Components and functions	Cluster analysis

To analyze the sum of existing solutions, the analysis criteria are partly overlapping those defined for individual architectures. The identification of degrees of freedom, restrictions, and potentials are explicitly defined for the sum of architectures, i.e. the solution space at this point, while the modularity follows the same aim as for individual architectures.

The identification of **degrees of freedom** is conducted on the basis of the number of alternative paths for the depicted functional requirements within the function domain. If the identified paths for a distinctive functional requirement are numerous, a solution can be chosen from a number of possibilities. On the other hand, if only one path is available, there is no degree of freedom, according to prevailing knowledge, and the structure for the respective function is set, if the function is required. This is also valid for the identification of **restrictions**. If numerous paths for a function are available, yet all have one element or interrelation in common, the element or relation is set and cannot be ignored during system definition. Bridge nodes and edges, as well as clustering and path analysis, are the structural characteristics that come into question for the analysis of restrictions, while the number of alternative paths is most relevant for the identification of degrees of freedom.

Potentials for improvement can be identified on the basis of the harmful functions, for example. Other functions or components that cause many harmful functions in different scenarios might be considered for revisions, while, on the other hand, a harmful function with multiple causes might be considered to turn into a positive side effect. The active- and passive-sum of components, functions, and harmful functions can be used for that cause.

The **modularity** of the solution space can be viewed differently among individual solutions. If modules show within the sum of architecture solutions, those might be considered as part of the product family, yet maintaining enough degrees of freedom for the surrounding architecture to realize different architectural concepts using that module. Benefits might appear during development, leaving certain boundary conditions open for decision-making, or, on the other hand, using modules to realize different architectures within the product portfolio or family. An example is serial and through-the-road hybrid solutions that use exactly the same electrical drivetrain.

The results of analysis carry different meaning for the different use cases resulting from the initial situation analysis. The importance of drive cycles and detailed circumstances was repeatedly underlined, and thus will be incorporated into the final evaluation. Selected results are shown in the following graphic, displaying the potential of the presented approach. There is insight into a few details, which are the cornerstones of the following solution search. Focus is placed on the results of the analysis based on the sum of architectures, rather than individual architectures, for which a property comparison based on listed results is sufficient.

The results of the architecture comparison deliver a heterogeneous picture of the strengths and weaknesses of architectures and components. As a major insight, the analysis cannot give preference to the architecture models on the basis of the architecture information alone. The scenarios in which the vehicles are to be moved are inevitable. The analysis of harmful functions underlines the hypothesis that thermal waste energy appears to posses major potential, since in all architectures and at numerous points, thermal energy results as a non-used energy output of components. Secondly, the harmful function “thermal loading” appears to be the harmful function with the most causative input functions.

Based on the analysis of the sum of solutions, the degrees of freedom for embedding the engine and motor/generator turn out to be the major advantage for the systematic architecture variation, since actual solutions deliver a number of possibilities. The modularity analysis shows clearly that the functional and physical modules differ, yet that a definition of modules is possible across different architectures. As a restriction, the drive shaft and axle gear connection turns out to be a common part of all architectures, originating from the fact that analyzed architectures did not include options with wheel hub motors, which would render axle gears redundant.

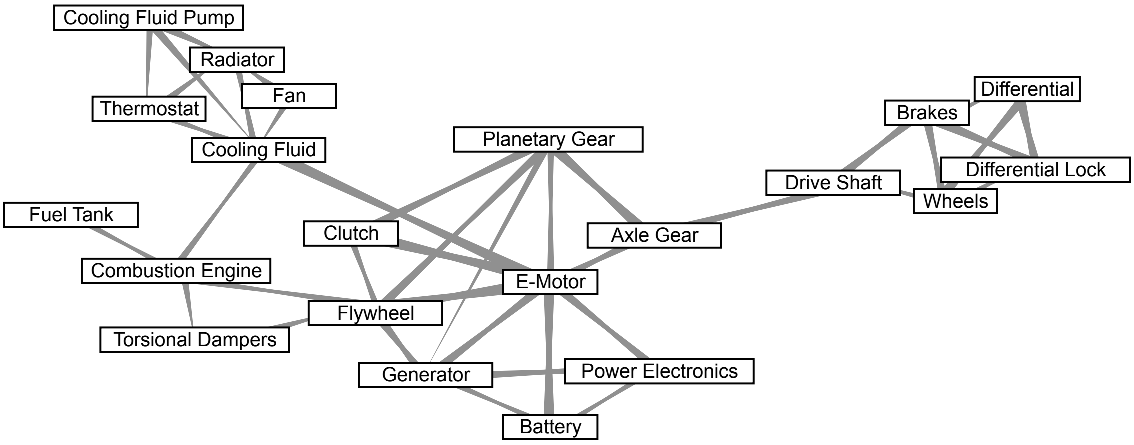


Figure 8-11 Graph depiction of ∑-DSM of components

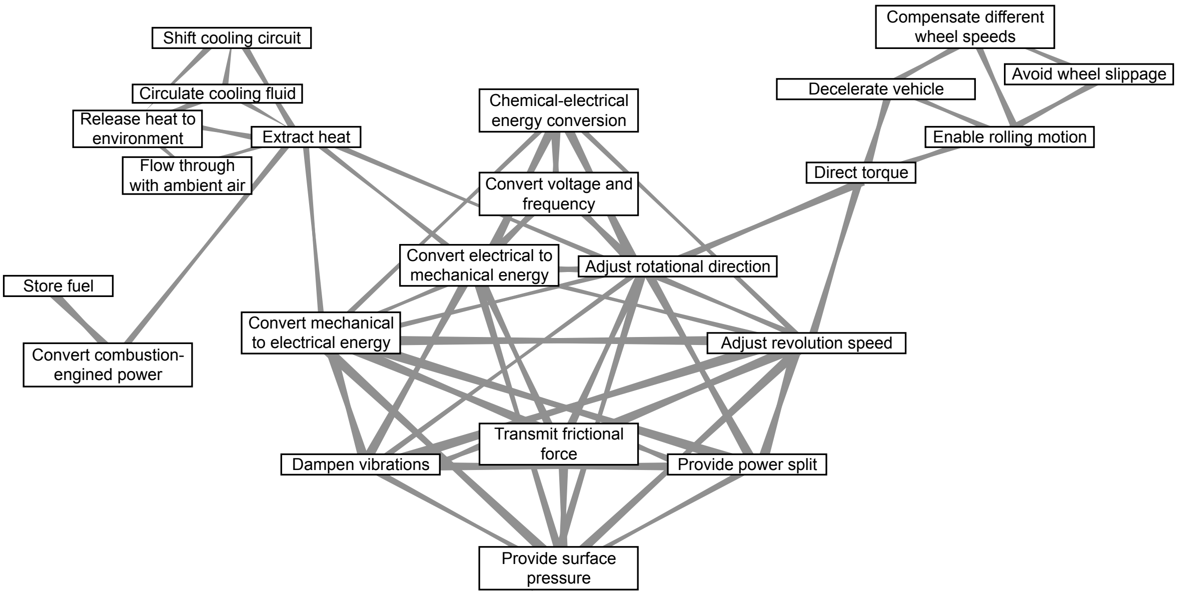

Compensate different wheel speeds
Shift cooling circuit
Avoid wheel slippage
Decelerate vehicle
Circulate cooling fluid
Chemical-electrical energy conversion
Release heat to environment
Extract heat
Enable rolling motion
Flow through with ambient air
Direct torque
Convert voltage and frequency
Convert electrical to mechanical energy
Adjust rotational direction
Store fuel
Convert mechanical to electrical energy
Adjust revolution speed
Convert combustion-engined power
Transmit frictional force
Dampen vibrations
Provide power split
Provide surface pressure

Figure 8-12 Graph depiction of ∑-DSM of functions

On the side of functions, the cooling-, fuel-, and drive-side-system are identifiable as modules, while the conventional and electric drive-systems are strongly overlapping, since many architectures strongly integrate the electrical part with the conventional part. Although the drive-side of existing hybrid architectures is highly integrated, the graph clearly shows that the cooling system of current architectures is not set up to absorb (or reuse) all of the waste heat of the vehicle. The independence of the cooling side from the electrical drivetrain shows also in the component matrix.

The above discussions show valid examples of structural analysis criteria for the presented case study. The presented examples naturally come as no surprise, yet indicate that the results are valid. While in previous chapters (compare chapter 5.4.3 and 5.6.3), a larger number of characteristics and possible analysis and evaluation criteria are mentioned, a reasonable choice and planning has to be conducted as to how each criterion supports the solution finding process. The automated conduction of a large number of analyses can be equally conducted with respective processing power, yet the human interpretation of results is the truly time-consuming aspect of the procedure. The meaning and interpretation are largely dependent on the respective use case and project.

The given examples have shown that structural characteristics have to be adapted and set into context by experienced users to cover eventualities and allow for the comprehensive establishment and analysis of a system model.

8.1.6 Validation

For the presented use case of drivetrain development, numerous aspects that were not demonstrated by the given example are to be considered. Those aspects, which make the development of such complex and dynamic systems possible in the first place, include the control strategy, drive cycle, detailed properties such as engine characteristics or battery characteristics, basic and environmental conditions etc. Still, the statements and results above provide directions and value as far as possible on such abstract level.

To validate the analysis results of the presented case study, two approaches were chosen. On the basis of system and component properties, the results can be evaluated through comparison to existing simulation results and other sources of hard figures about the respective architectures. Relative comparisons of efficiency and structural complexity reflect in the results of simulations and modeling efforts for simulation. The hypotheses resulting from the structural analysis were confirmed by means of the mentioned measures of validation.

For the results concerning the overall architecture structure, interviews with experts were conducted, based on the retracing of findings into the source models. Based on known representations, involved experts can evaluate and discuss results more easily, while the MDM model provides the overview and comprehensible origin of the analysis results.

8.1.7 Implementation

The implementation step includes the drawing of conclusions based on the analysis results. In cases of small and less complex systems, implementation may include the synthesis phase, if analysis results are unambiguous and tasks clear.

First, the validated analysis results confirm parts of the original objectives of the case study. The drivetrain concepts were functionally and physically analyzed, the domains decomposed, and properties for elementary components or building blocks determined. Evaluation criteria for existing solutions were derived, and their reliability and suitability approved. Thermal waste energy was identified as a potential type for energy recovery or reuse from a structural perspective. The structural characteristics derived from the analysis of the functional and physical systems could not point to an immediate demand for action, since the restrictions were agreed to be inevitable concerning the current state of the art. Nevertheless, synthesis aims to tap the full potential of actual architectures and possibilities.

The remaining tasks for the synthesis steps are, in detail, the solution finding for the thermal energy recovery, the search for alternative solutions of drivetrain concepts on levels of decomposition, and the evaluation of concepts, including modularity, which should conclude the synthesis phase. The following figure provides the outline for synthesis, based on the identified domains in the meta-model.

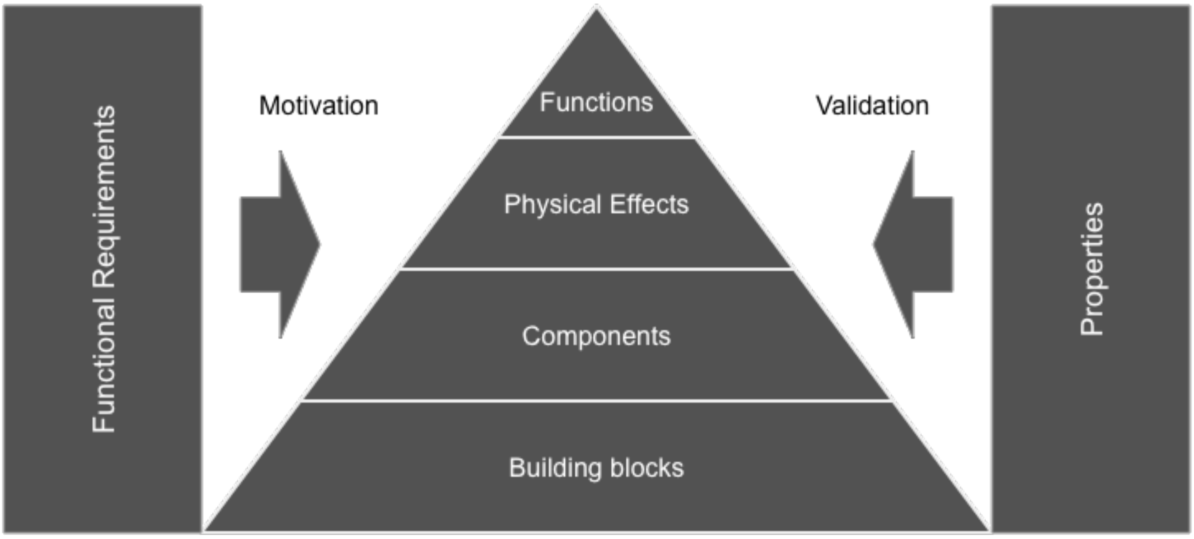


Figure 8-13 Context of the architecture framework for synthesis

The following sections will identify the respective domains for the remaining synthesis tasks, and provide solutions utilizing the methods outlined in previous chapters. The following step of solution search focuses on the physical entity domains. Solutions to complete the solution space will be sought within the domains of components and physical effects. The “building blocks” represent the domain of assemblies. Assemblies in the context of the case study are understood as groupings of components, which as a group form a functional unit. Examples for a functional unit are the functional requirements. An assembly would thus fulfill one or more functional requirements. The systematic exploration of the solution space will demand the consideration of both the functional and physical domains. A systematic variation of the functional domains might, in return, demand revisiting of the task of solution search if new solutions are required. The iterations were explicitly discussed and underlined in chapter 7.4.2. The following discussions will nevertheless be structured in a task-based manner to clarify each step with examples.

8.1.8 Solution search

The task of solution search aims for the completion of the solution space, i.e. the identification of potential solutions to (sub-) problems of the system. In the given case study, the sub-problems were identified by the component and functional decomposition (see chapter 8.1.2).

The solution search to elaborate the solution space was conducted on the basis of literature and competitor research. Numerous variants of components can be found, which fulfill the defined functionality. The solution search can be conducted on three levels, i.e. the functional class level, the principle level, and the specification level. Among others, a combustion engine, for example, can fulfill the function “convert chemical to mechanical energy”. On the principle level, a differentiation can be made between a Diesel- and Otto-engine. Both can be detailed on the specification level, according to their number of cylinders, cylinder capacity, power, engine characteristics, maker, etc.

The research of possibilities and potentials can be conducted on different levels, i.e. functional class, principle, and specification levels. For the presented case study, the specification level was chosen to find a critical number of representatives to confirm the properties on principle and class levels. At the same time, the functional classes and principles were critically challenged and research was carried out to determine whether different principle solutions exist that fulfill the functional classes’ main properties. The questioning of the functional class level was conducted during the systematic exploration of the solution space, since then the interactions between different domains and classes of domains show and are required for a comprehensive solution finding process.

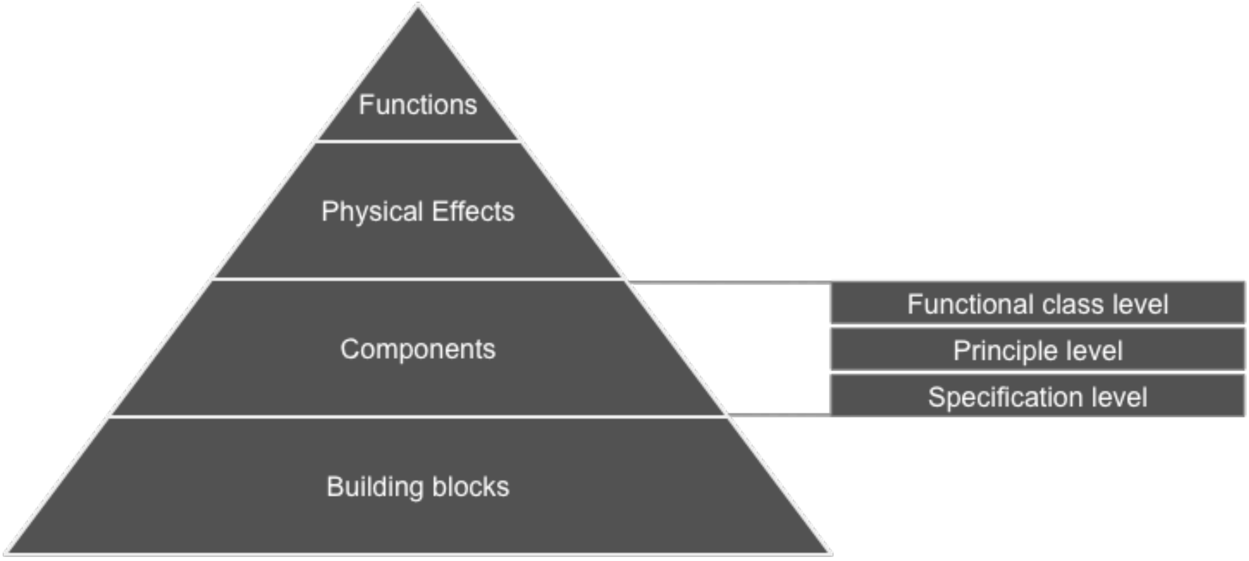


Figure 8-14 Scope of solution search

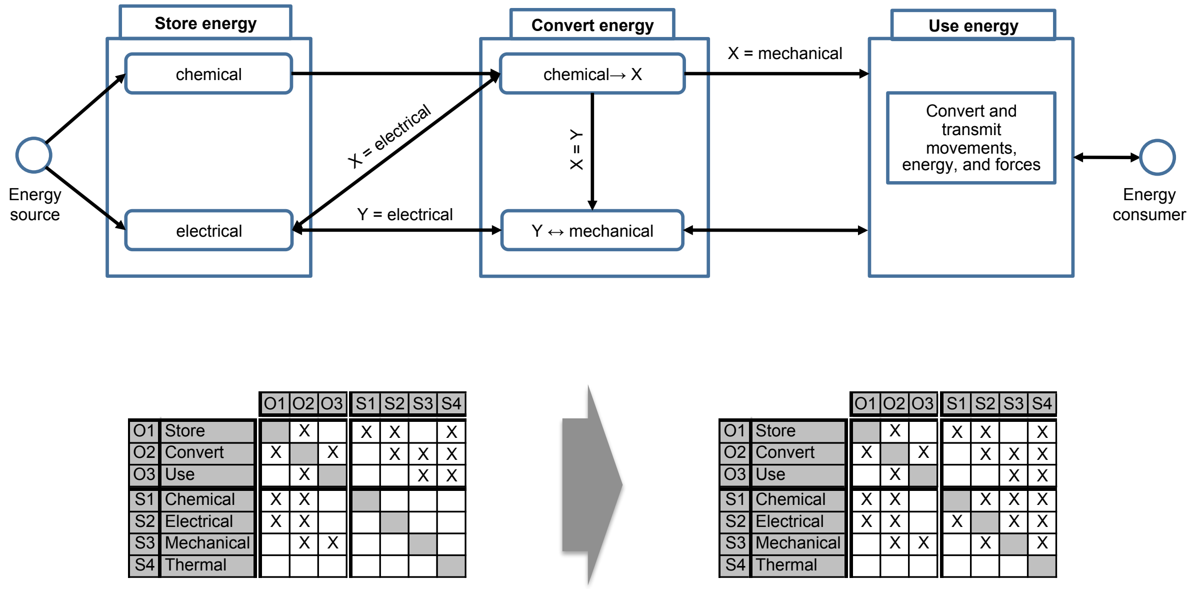
The solution search focused on the system decomposition and potential solution elements in the context of the functional fulfillment of requirements. The objective of energy recovery will serve as an example in the following section, which discusses the systematic exploration of the solution space.

8.1.9 Systematic exploration

The systematic exploration of the solution space was conducted on two levels, i.e. the exploration of the functional architecture, as well as the physical architecture. Each of these two directions can be decomposed as follows: the exploration of the functional architecture is conducted on both the level of primary and secondary functions, while the exploration of the physical architecture is conducted on all levels identified within the meta-model for the case study, i.e. assemblies, components, and physical effects. Therefore, the assemblies and components are chosen to enable variations on architecture level, while the synthesis on the level of physical effects and components was conducted to identify variations on component level.

- Exploration of the functional architecture
    - On the primary functional level
    - On the secondary functional level
- Exploration of the physical architecture
    - On the level of assemblies
    - On the level of components
    - On the level of physical effects

The following sections will introduce several models and methods that were applied for each of previously mentioned synthesis cases, discussing the outcome, as well as the observed advantages or disadvantages.



		O1	O2	O3	S1	S2	S3	S4
O1	Store	▨	X		X	X		X
O2	Convert	X	▨	X		X	X	X
O3	Use		X	▨			X	X
S1	Chemical	X	X		▨			
S2	Electrical	X	X			▨		
S3	Mechanical		X	X			▨	
S4	Thermal							▨

		O1	O2	O3	S1	S2	S3	S4
O1	Store	▨	X		X	X		X
O2	Convert	X	▨	X		X	X	X
O3	Use		X	▨			X	X
S1	Chemical	X	X		▨	X	X	X
S2	Electrical	X	X		X	▨	X	X
S3	Mechanical		X	X		X	▨	X
S4	Thermal							▨

Figure 8-15 Exploration of the functional architecture on the level of primary functions

For the **variation on the primary functional level**, the decomposition into operations and states was chosen, enabling the systematic exploration of possible scenarios on the core functional level. The above figure depicts the flow-oriented functional model on the primary level on the top. The matrix depiction (lower half of the above figure), as one of its major benefits, allows for a clear overview of what is and what is not considered in the current solutions. White spots in the overview can be systematically questioned and intentionally ruled out or reconsidered. Deliberately, the thermal output energy was introduced for the matrix depiction, filling the obvious gap in the diagram depiction (which focuses on the main energy flow). The DSM and DMM matrices of the model can be separately analyzed. For example, the input states to operations DMM on the bottom left clearly show that:

- Mechanical energy storage was decided to not be a practical option (although examples for flywheels in the automotive sector exist),
- The direct use of chemical or electric energy is not possible (a conversion to mechanical energy is necessary),
- Thermal energy is currently not used as an input for any operation.

As a conclusion, the use of thermal energy can be discussed and incorporated accordingly into the model. Therefore, a systematic exploration is necessary, whether concepts for “thermal-electric”, “thermal-mechanical”, or “thermal-chemical” converters exist on the component- or physical effects-level. The domain-mapping logics enable the derivation of dependencies and matrices not originally part of the model, such as the DSM of states on the bottom right of the depicted MDM.

The **analysis of the secondary functional level** results in the most relevant harmful and secondary functions, yet cannot point to significantly new results. The path analysis on the functional level for the fulfillment of functional requirements delivers few solutions. The chosen functional level thus does not deliver more valuable input for the exploration of the architecture solution space than the results from analysis (chapter 8.1.5) have shown.

The **architecture variation on level of assemblies** implies the systematic combination of building blocks on the principle level, which partly overlap and were identified on the basis of the results of analysis. Additional building blocks, for example resulting from the identification of thermal energy recovery, widen the solution space and the number of possible combinations accordingly.

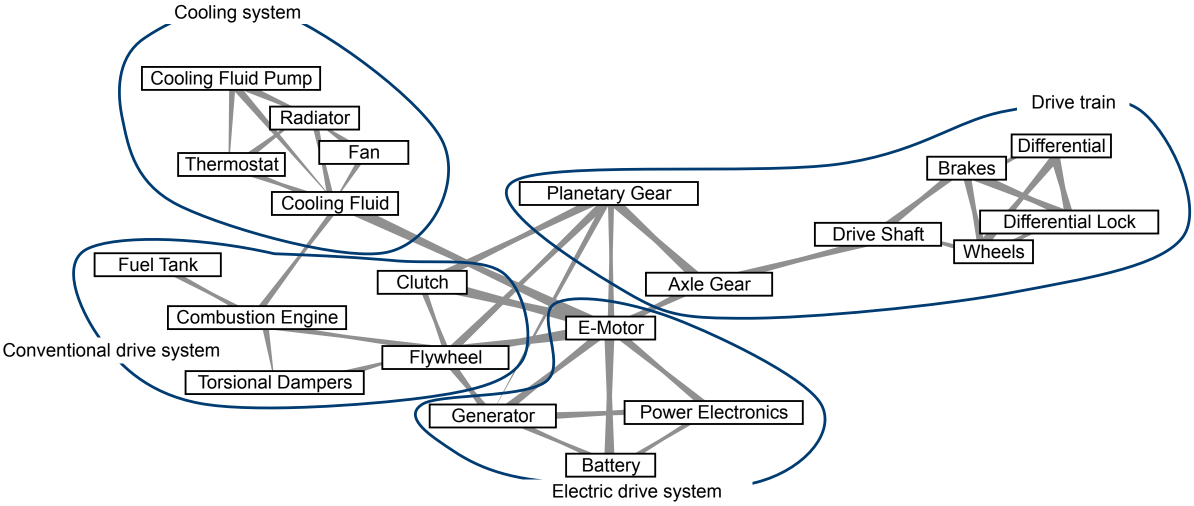


Figure 8-16 Identification of building blocks within the solution space (example of component structure)

The building blocks can be evaluated, for example depending on the degree of efficiency with which the respective functional requirement is fulfilled, the number of entities required, or the resulting modularity of the architecture. To identify the building blocks, a complete architectural depiction of the solution space is required, as will be presented in the following section.

Since the building blocks result from the analysis of the known solution space, more building blocks or architectures can be defined by the **systematic combination of components**. Typically, the decision tree or the compatibility matrix are the common methods to define the combination of components, based on a defined set of components. The following figure depicts the cutout of a possible *decision tree* with relevant decisions for the structural combination of drivetrains.

The starting point for the decision tree of the case study, and therefore common to all variants, is the following collection of properties: a combustion engine as part of the drivetrain, a gearbox (manual or automatic), and the premise for all branches of the tree is to define autarkic architectures with the combinations given with the variant tree.

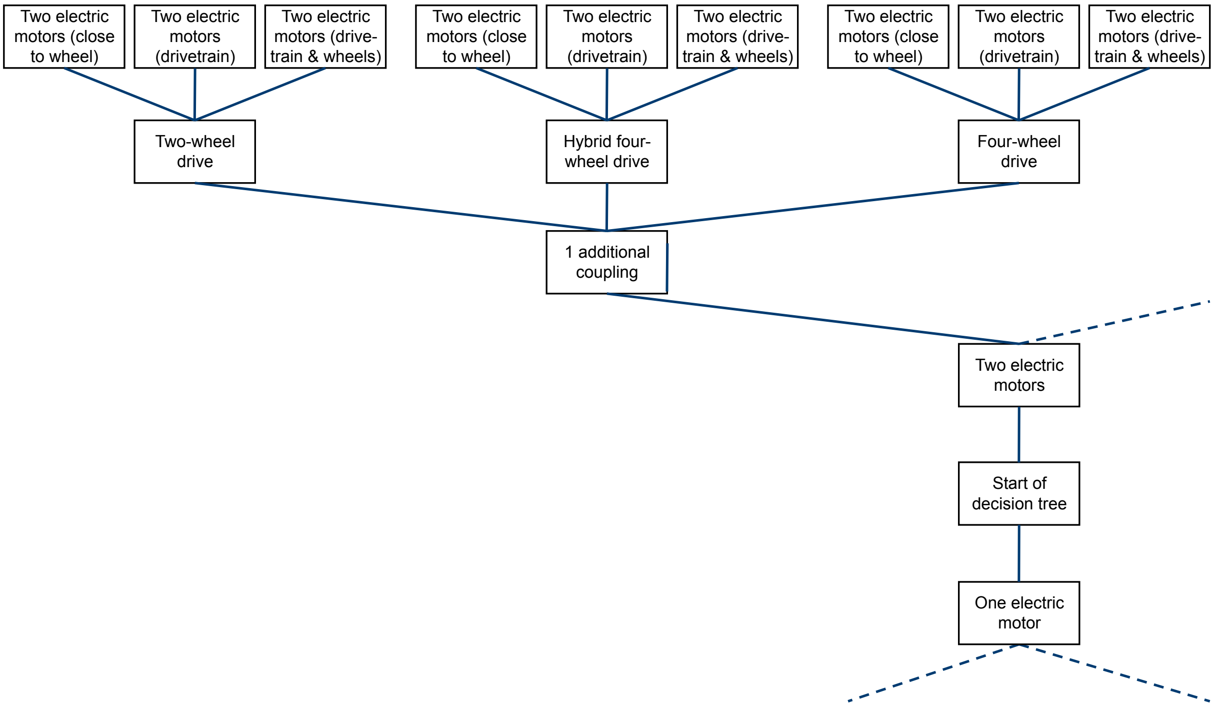

Two electric motors (close to wheel)
Two electric motors (drivetrain)
Two electric motors (drive-train & wheels)
Two electric motors (close to wheel)
Two electric motors (drivetrain)
Two electric motors (drive-train & wheels)
Two electric motors (close to wheel)
Two electric motors (drivetrain)
Two electric motors (drive-train & wheels)
Two-wheel drive
Hybrid four-wheel drive
Four-wheel drive
1 additional coupling
Two electric motors
Start of decision tree
One electric motor

Figure 8-17 Decision tree for the case study (example)

The second level of the variant tree differentiates between architectures with one or two electric motors, followed by the differentiation of drivetrains with no or one coupling (additionally). Further down the tree, there is a decision to be made between two-wheel drive (front or rear), four-wheel drive and hybrid four-wheel drive (for example, one axle conventional, one electrical). The final part of the decision tree is then to differentiate between three positions of the electric motors. If there are two electric motors, the positioning of the motors is divided into three different cases: the first position is the wheel hub motor (or any other form of motor close to the wheels) for one axle, with the second motor integrated in the drivetrain; the second position is a wheel hub motor for two axles; or as a third position, there are two motors positioned along the drivetrain. For branches with one electric motor, 12 possible combinations exist (the positioning of the electric motors allows for only two different positions), while for branches with two electric motors, 18 possible combinations exist, resulting in 30 mathematical possibilities overall, of which six do not lead to reasonable architectural results.[94] To evaluate the reasonability of the decision tree, selected existing hybrid concepts were compared with the branches, depicted in the following table.

[94] Based on an earlier, more detailed version of the decision tree, the differentiation was made between autarkic and plug-in hybrid, five different gearboxes and five different energy converters, as well as two energy storage variants. The positioning of the electric motors was combined with the two- and four-wheel-drive, resulting in four more variants on that level, leading to a decision tree of 1600 variants.

Table 8-6: Comparison of existing hybrid concepts to the resulting branches of the decision tree

Hybrid concept	Electric motors	Additional coupling	Drive concept	Electric motor positioning
Peugeot RC HYmotion4	1	0	Hybrid 4WD	Close to wheels
Peugeot Prologue HYmotion4	1	0	Hybrid 4WD	Close to wheels
Citroen HDI Hybrid	1	1	2WD	Drivetrain
Citroen C Metisse	1	0	Hybrid 4WD	Close to wheels
Audi A1 Sportback Concept	1	1	2WD	Drivetrain
Land Rover Diesel ERAD	2	1	4WD	Close to wheels
Citroen C-Cactus Concept Car	1	1	2WD	Drivetrain
HHF Hybrid Concept Car	1	0	Hybrid 4WD	Close to wheels
Peugeot 308 Hybrid HDI	1	1	2WD	Drivetrain
Porsche Cayenne Hybrid	1	1	4WD	Drivetrain
Saab Bio Power Hybrid Concept	2	1	Hybrid 4WD	Drivetrain/ Close to wheels
Touran Eco Power	1	1	2WD	Drivetrain
Fiat Multipla Hybrid Power	2	0	2WD	Drivetrain
X3 Efficient Dynamics	1	1	4WD	Drivetrain
Audi Metroproject	1	0	Hybrid 4WD	Close to wheels

The decision tree can propose the ingredients for a drivetrain, yet cannot pose the architecture of the chosen entities. This is especially true in the case of the drivetrain, where numerous entities possess a number of possible interrelations with other entities. The following figure depicts two different architectures, both based on the same decision tree. Naturally, not only the decisions for entities, but also components can be part of a decision tree. Therefore, almost all possibilities have to be predefined, making the decision tree a medium for visualization, rather than synthesis. The following figure depicts two solutions for one branch of the decision tree.

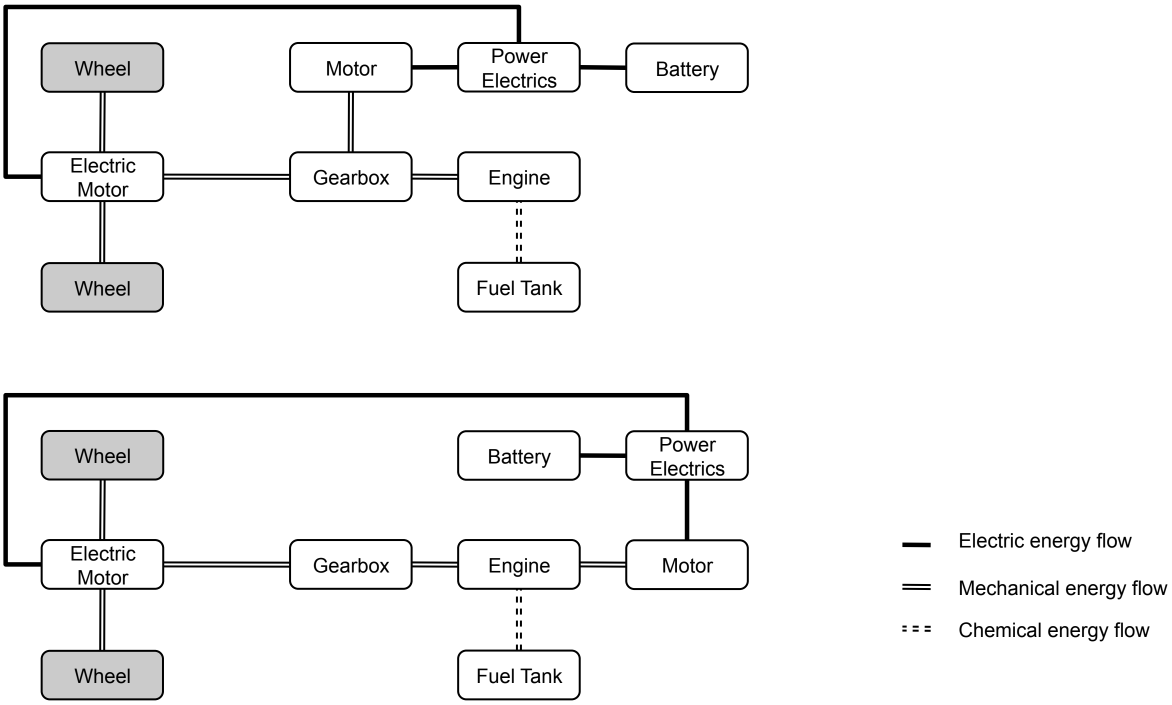
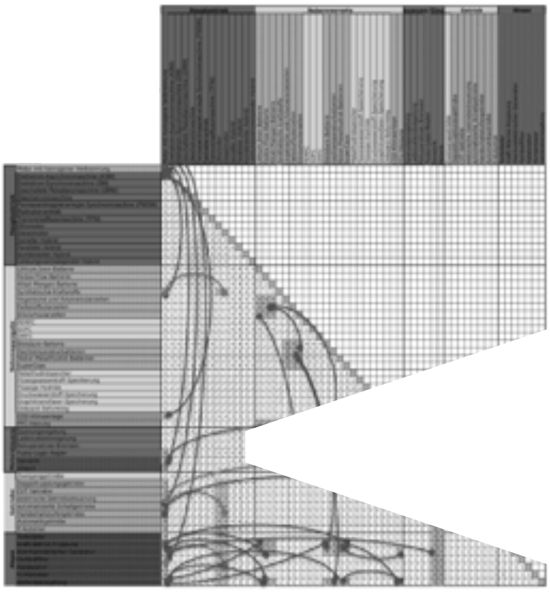


Figure 8-18 Different structural solutions resulting from one branch of the decision tree

The *compatibility matrix* depicts the physical entities of the product architecture and their compatibility with one another, i.e. the constraints and exemptions in a variant management application. In contract to the previously mentioned applications of matrices, the compatibility matrix does not depict the actual interrelations between entities, but rather their general compatibility. The matrix depicts which entities are generally allowed in one configuration of the product architecture, yet cannot visualize how the components are connected or interrelated. The results of the compatibility matrix are similar to those resulting from the application of the decision tree or the morphological chart. All of these approaches share the depiction of a general compatibility of the entities, without regarding the actual structure of the architecture.



1 – Combination of solutions not possible
(technologies are incompatible)
2 – Combination of solutions not reasonable
(no benefits but redundant functions to be expected)
3 – Neutral combination
(no redundancy but no imminent effect)
4 – Reasonable combination
(combination not imperatively required but beneficial)
5 – Mandatory combination
(combination of technologies unavoidable)

4	4	2	2
1	1	1	1
4	4	2	2
3	3	4	4

Figure 8-19 Compatibility matrix for the case study: overview (left) and combinational logic and examples (right)

The *depiction of the solution space,* as proposed in chapter 6.5 intends to provide exactly the interrelations between entities missing from the approaches above. The comprehensive depiction of the solution space alone allows for the identification of building blocks, as previously discussed. In additional, the overall solution space can be analyzed, e.g. via cluster analysis, to identify modularity or constraints for modularity among the available solutions, to intentionally decide for or against interfaces between modules. The following figure depicts two possible definitions of interfaces, based on the differentiation of an integrated or separated electric drivetrain (through-the-road vs. parallel hybrid). If possible, based on the defined solution space, solutions can be deliberately defined as modular. Objectives and further circumstances for the architecture and its evaluation define the importance of the modularity criterion, yet in many cases, modularity poses one of the major drivers for product architecture definition.

The systematic variation on the level of **physical effects** is not the first step to take when defining architecture variations. The use of physical effects has its place and origin in design methodology for the definition of solutions, on the basis of functionally described products. For the systematic variation of product architectures, physical effects can be applied to cope with constraints that cannot be dissolved by current solutions and the means discussed in preceding sections. A precise problem description can usually be defined on the basis of the constraint to be solved; in the use case, this is the recovery of thermal energy. As such, the resulting objective is comparable to a typical design, rather than an architecture problem.

Existing approaches aim for the identification of individual effects to fulfill functions [Lauer et al. 2008, Ponn & Lindemann 2008]. For more complex problems, a combination of effects is required to fulfill the desired function. Important for the successful application is the identification of the desired input- and output-variables. Similar to the depiction of the solution space on the level of components and functions, the solution space within the physical effects domain can be depicted accordingly, based on existing libraries of physical effects. The comprehensive depiction enhances existing approaches, pointing out chains of effects.

The following figure shows a network of physical effects coupled with energy states, i.e. the input- and output variables of effects. Given the objective of reusing thermal energy, the effects network shows numerous possibilities to transfer thermal energy into other desired energy states. A systematic analysis of (short) paths between desired energy states points to potential solutions, usually covered by existing technologies or components. The search for existing solutions on the level of components is naturally also valid for the given objective.

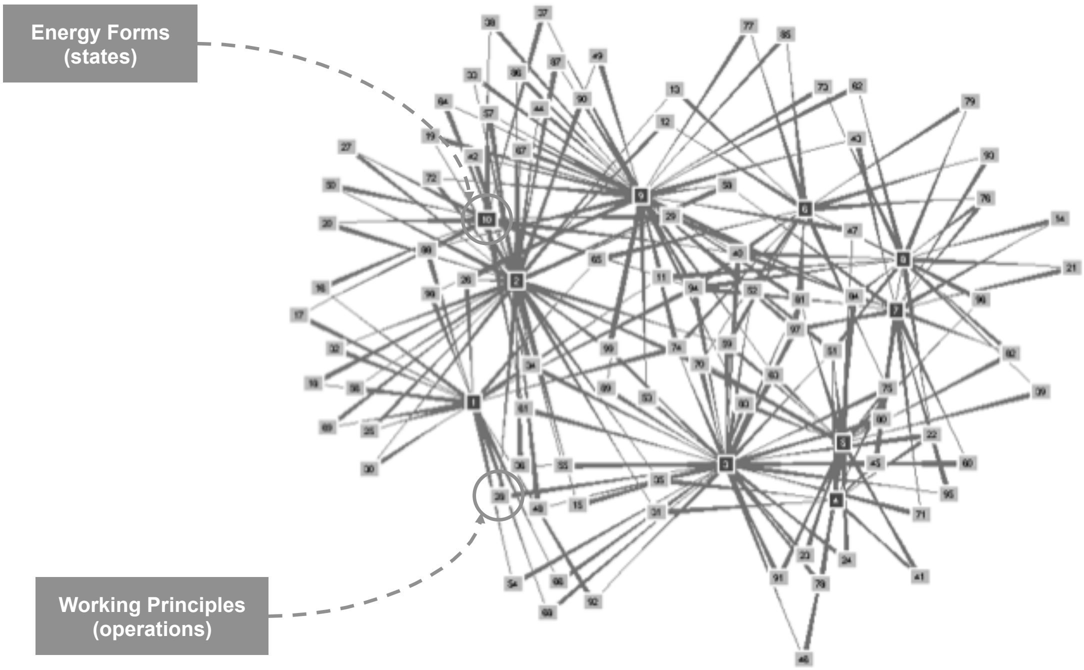


Figure 8-20 Depiction of solution space in graph form (from DEUBZER & LINDEMANN 2008)

The above sections introduced a number of possibilities for the systematic exploration of the solution space, on the basis of the product architecture model. The most important benefit for the presented task is the search for solutions within the different domains and on different layers of abstraction within the model.

8.1.10 Evaluation and decision

The systematic exploration of the solution space in the previous step was conducted on a number of different levels. The evaluation and choice of entities, from either the functional domain, assemblies, components, or physical effects, behaves similarly. However, for both the exploration and evaluation, the interplay of domains has to be considered. Novel elements to the solution space open possibilities within related domains. Physical solutions new to the solution space, for example, open up new functional possibilities or evoke different properties of the overall system.

Accordingly, the evaluation of properties has to be conducted on all levels of the physical solution space, in order to grasp the impacts between domains. If the properties of components have changed, the behavior and properties of assemblies and/or the architecture are likely to have changed equally. The evaluation criteria are determined by approaches defined during analysis.

The evaluation criteria for **architectures** were introduced in chapter 8.1.5 on analysis of the case study. Therein, the evaluation of existing architectures was conducted, based on the following criteria. These, in turn, can be applied to evaluate newly developed architectures during the synthesis phase.

- Structural complexity
- Number of user-functions
- Properties (degree of efficiency)
- Modularity

The importance of each of the criteria and their weighting against one another can be conducted based on scenarios and/or use cases, referenced at the end of this section.

On the level of **assemblies**, the functional building blocks in the case study, the degree of efficiency was the main concern for evaluation. Additionally, the criteria defined for the architecture were important considerations. The structural complexity and number of user-functions are equally applicable, for example, while the modularity of the building blocks requires the architectural context. The number of interfaces can give an idea of the modular properties of each building block. It is important for all of the domains evaluated that the evaluation not be considered in solely one domain. Requirements and/or properties are reflected in the different levels of the architecture; thus, evaluation must be considered on all levels accordingly.

On the level of **components**, the properties were defined by the example of electric storage components. These can provide a ranking of components with similar functions, but again the context within other levels is important for the comprehensive evaluation. The evaluation criteria for components, if not generalized on the level of cost, weight, etc., differ between classes and case studies, and as such must be defined based on the objectives and circumstances of the project. Numerous possibilities were identified in chapter 5.6, which also provides potential methods by which the evaluation can be conducted sensibly.

The rating of **physical effects** was previously not considered. In general, the evaluation or rating of physical effects (or technologies, i.e. novel combinations of effects without explicit context in products) is hardly practical. The context in which the physical effect is used has a significant impact on the evaluation of the effect. In fact, physical effects can be rated exclusively in the project- and product-contexts. In the following points, examples of evaluation criteria for physical effects that were relevant for the presented case study are provided. These can be adapted for further use in other projects and show the limited possibilities of the evaluation of physical effects.

- *Environment and boundary conditions:* In the context of the automotive drivetrain, effects have to be evaluated, whether they are dependent on certain boundary conditions or environmental properties, and if those conditions are prevailing or educible within the automotive context. Effects that are dependent on extremely low temperatures, for example, are difficult to handle (e.g. the challenges concerning the liquid storage of hydrogen).
- *Threshold values:* A significant concern is whether the physical effect allows for energy turnovers of suitable amplitude for the automotive drivetrain development. Physical effects capable of only minor forces are likely irrelevant for the presented case study, if they cannot be scaled or multiplied (which is the case for accumulator technologies, for example). The possibilities of scaling and multiplication of the effect have to be considered before excluding certain effects.
- *Material:* The required choice of materials for the physical effect points to disqualifying criteria for certain use cases. In the given context of the case study, highly toxic or radioactive materials are likely not practical for the use in an automotive context.
- *Tolerances and accuracy:* Physical effects often require tolerances to be within small margins to be fully functional. These require not only appropriate production technologies, but also the adherence to accuracy during use. The practicability of effects needs to be controlled under those concerns, including the aspects of cost and durability.
- *Degree of efficiency:* The degree of efficiency as an evaluation criterion was considered for the evaluation of other physical domains within the product architecture framework. Physical effects themselves can rarely be evaluated without the context of surrounding technologies and the product itself. However, in some cases, the principle capabilities, i.e. the potential of efficiency of certain effects, might already provide a disqualifying criterion for the use of the respective effect.

To be able to set the aforementioned evaluation criteria of all domains into a meaningful environment, the identification of scenarios or use cases is inevitable. Differentiated scenarios allow for the weighting of criteria with respect to each scenario. As a result, the entities of the architecture can evaluate if the required focus and target of the architecture is met. For the presented case study, a major scenario was derived, within which two distinct use cases and thus two distinct evaluation grids can be defined. With the boundary conditions and anticipated development of the automotive environment provided in chapter 8.1.1, two possible use cases can be briefly described as “City vehicle” and “Multi application sedan”. While the city vehicle can be described as a small vehicle mainly used for short paths, stop and go and potentially within zero emission zones in the near future, the multi application sedan requires a large interior space and different applications, ranging from occasional city rides to outer city and long distance traveling. Based on the differentiation of the use cases depicted in following table, suitable drive cycles can be derived and a weighting of criteria conducted.

Table 8-7: Rough use case outlines for the case study

City vehicle
Short distance traveling
Potential zero emission zones
Small two-person vehicle
Drivetrain complexity and space critical
Homogeneous applications
Small and homogeneous product family
Low price segment

Multi application sedan
Mixture of path lengths
Small amount of zero emission zones
Large family space sedan
Drivetrain complexity and space less critical
Heterogeneous applications
Large and diverse product family
Medium price segment

To give an idea of how the use cases impact the weighting of evaluation criteria, a few examples support the understanding of the mapping process. The character of traveling distances and diversity gives the outlines for the importance of the different driving modes. While electric driving is within the focus of a city vehicle, the driving modes for a multi application sedan tend to be equally relevant relative to one another. Accordingly, the number of available driving modes is of higher importance for the multi application vehicle, compared with a drivetrain designed for single purposes. The restrictions of space and cost point to rather low complexity solutions for a city vehicle, while the comprehensive product family, price segment, and available space of a sedan allow for a more elaborate solution and increase the importance of the modularity of the architecture. While the given use cases allow for the differentiation of architectures at a high level, the evaluation criteria for the identification of suitable physical effects and/or components are strongly dependent on the derivation of requirements, based on the architecture surroundings.

Of course, the architectural decisions cannot be made based on the above-introduced measures alone. The application of the case study was not limited to energetic-functional aspects of the structure. The comprehensive evaluation requires complex vehicle dynamics simulations, operational control strategy for engines, gears, and overall energy management, detailed component properties, drive cycles, virtual drivers, etc. The architectural considerations as discussed complete the comprehensive picture, give insights into where to set focus on architectural decisions, and complete the solution space.

8.2 Discussion

The introduction of the product architecture management approach allows for the systematic analysis of architectures and search for solutions in the context of complex architectural problems. The real-world objectives of the problem were reduced to the architectural measures for the case study. The application showed the potential, based on practical examples.

The application of the **architecture management approach** showed the possibility of applying different methods, such as functional models, structural analysis, and evaluation. Synthesis was systematically conducted on different architectural levels and different domains, spanning a comprehensive solution space. An overview of the solution space was given on different levels and the impact between domains and levels of abstraction was presented.

The product **architecture model** makes results and properties intuitive to grasp, and allows for the mathematical accessibility, as well as the interrelating of existing models, both important elements for analysis. For the information acquisition, the model enables a sharp definition of entities and relations, and guides the process of acquisition.

The **product architecture framework** enables the comprehensive establishment of situation analysis and structured information acquisition during the project setup and within the first tasks. The spanning of possible classes of domains and domains the framework supports a comprehensive overview from the start.

Limitations especially show in dynamic systems, where behavior and properties can only be deduced using dynamic simulations. Altogether, the introduced measures can only be understood as a support during the early phases. They point to promising directions of the architecture and combine evaluation criteria and properties, which cannot be provided through other means.

9. Discussion

The presented work covered numerous aspects of the management of product architectures. Starting with the role and evolution of the relevance in industry and reflecting on the suitable approaches in science, the work systematically identified the major challenges for coping with product architectures. Approaching the nucleus of the challenges, aspects of complexity in general, and in engineering design of complex products in particular, were demonstrated by means of different schools of thought. To comprehensively cope with complex architectures, the existing methods and approaches were incorporated into a three-pronged approach, based on the Multiple Domain Matrix approach. The three pillars of the approach include a framework regarding the content, a model capable of handling the complex system information, and a procedural model for the systematic coping with complex systems. The concluding sections of the work will sum up the findings and remaining shortcomings. The outlook section points out potential and promising directions for future work on the subject.

9.1 Conclusions

The following summary will discuss the main findings of this work and build up to the challenges remaining for the future. The first descriptive study discussed the **role of the product architecture,** based on a literature review. From the perspective of customers and markets, the resulting complexity and challenges in engineering design reflect on the product architecture. A reasonable handling of variant-rich product architectures with multilayered requirements and differentiated perspectives is required. The markets, as well as the comprehensive lifecycle perspective, are the main causes for this diversity. The different aspects of organizational complexity result in implications on the product architecture and vice versa. Team definition, multiple project environments, knowledge and decision-making, as well as existing value networks, define the restrictions and boundary conditions for product architectures. The character of the engineering design process, its recursivity and iteration have to reflect in the models and approaches for the product architecture. Decision-making as a major property of the design process has to be supported by a comprehensive approach in product architecture management. All in all, the “complexity”, which is regularly referred to in the context of product architectures, was detailed, and its origins and implications were discussed. As part of the descriptive study, the **coping with complexity** in the context of engineering design was discussed. Different existing schools of thought and fields of research were characterized and their suitability for product architecture management in the early phases analyzed.

The first prescriptive study provided a **framework for systems architecting**, resulting from an intensive discussion of the modeling of product architectures, its requirements and possibilities, based on the findings of the descriptive study. At this early point in the process, the framework provided an outlook of the product architecture management approach, pointing to a feasible modeling approach on the one hand, and providing the entities or artifacts of the product architecture that are most likely to be relevant for the product architecture on the other.

To validate the framework and provide a profound scientific basis, the second descriptive study delivered a comprehensive **method review**, based on a literature review and mapped to the requirements for the management of product architectures identified in the previous chapters. As a result, not only were feasible methods identified, but the requirements to a solution were also summed up as: *consistency* (support to recursive and iterative procedures), *comprehensiveness* (consideration of different relevant entities on different levels of concretization and incorporation of stakeholder perspectives), and *flexibility* (modeling approach to a few existing methods and models, based on an adaptable procedural model).

Based on the identified requirements and boundary conditions, the second prescriptive study introduces novel solutions to the problem in two ways. First, **novel constituents** to the approach, missing from the review, are defined, including the modeling and interrelation of existing methods in MDM notation, the coping with hierarchies and recursive procedures, and finally the support of synthesis in general, based on the depiction of the solution space. Second, the provision of an **approach for the management of product architectures** is introduced, based on the architecture framework, model, and procedural model. The approach is designed to fulfill identified requirements and eliminate a number of shortcomings, uniting existing and novel methods and solutions.

The last descriptive study provides a **case study-**based example of the application of the approach for the management of product architectures. A number of cornerstones of the approach could be validated within the example, and the overall approach and combination of framework, model, and procedural model identified as feasible.

The **results** of the work have shown that the *demands for the management of product architectures* are apparent in industry and in science. The majority of current methods and approaches in design are not intentionally designed to meet the requirements for the management of product architectures. The Multiple Domain Matrix (MDM) is capable of providing a sound *backbone for systems architecting*, for which models and approaches are frequently demanded in literature, yet seldom provided. The framework and procedural model enable the *practical application of the MDM* for product architecture management, and the use of suitable methods within a coherent system and modeling context. Therein, the *gap between analysis and synthesis* was partly overcome and the *recursive and iterative character of the design process* was accounted for. Additionally, the search for solutions was enabled, not only as a sequential process, as is often proposed, but also in the sense of a *systematic exploration of the solution space* within and across different domains and levels of abstraction.

The **remaining shortcomings** of the approach show in different points. The introductory chapters showed the multiple and diverse demands for the management of product architectures. While the presented work could introduce an approach to cope with the core of the product architecture, i.e. the structure, numerous aspects exist which cannot be considered, portrayed or optimized by the approach presented, or at least could not be validated. As the case study showed, the consideration of dynamic aspects and system behavior are especially difficult at present, and the interfaces of the introduced approach to further means, such as simulation etc. are not yet clearly defined. Furthermore, the approach could not be comprehensively validated. The case study provided an example regarding the energetic-functional questions, but left out numerous other issues such as geometrical, manufacturing, variant management issues, etc. While those can be tackled through the means introduced, proof of this was not given. Although the conceptual approach could be validated with above-mentioned exceptions, its acceptance among practitioners could only be vaguely approached in conducted projects of the author. For the approach to be a valuable contribution to industry, the methods need to be understood and accepted.

9.2 Outlook

Based on the findings and shortcomings, the remaining potential for further development can be identified. The framework, model, and the procedural model allow for the definition of further means.

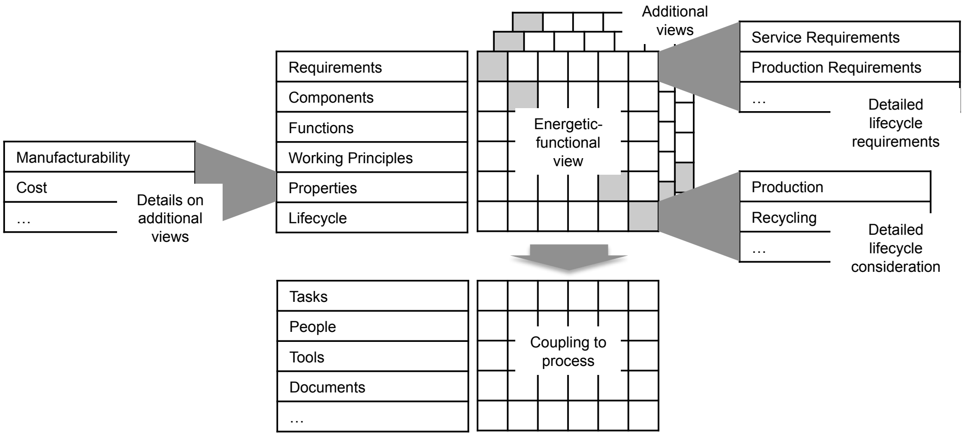


Figure 9-1 Potential extensions of the approach (examples)

First of all, the **framework** as introduced focuses largely and in detail on the known and strongly related entities of the architecture, i.e. components, functions, and working principles. The other classes of domains, such as requirements, properties, and lifecycle, were considered, but not in adequate depth, considering their importance. The requirements alone allow for their own framework of entities, not only by considering the lifecycle requirements stemming from e.g. service and production more comprehensively, but also by detailing and understanding the interplay of requirements and the architecture more comprehensively. The domain of properties was largely considered from a structural point of view, yet inherits numerous other aspects, such as cost, behavior, manufacturability, etc. Though difficult to grasp from the perspective of structural means, the domain of properties present important aspects for the decision-making process in the context of the product architecture. The same is effective for the lifecycle domains, which were only marginally considered. Production, recycling, or transportation need to be effectively coupled, yet indications are vague as to whether the presented approach can also cover the means of these domains. For the design process and business processes in general, the MDM approach has proved to be reasonable.

The considerations described in the **architectural model** and in the case study focus on structural and energetic-functional aspects of the architecture, yet additional views need to be validated as well. While the presented model is very powerful for the analysis and depiction of structural means, other approaches, likewise beneficial for the management of product architectures, need to be considered and coupled. Since the core idea of the approach is not to replace or contrast other methods, the integration of further means needs to be evaluated. For example, the rule-based synthesis can be based on the analysis results of the approach, since both are closely linked to Graph Theory. Apart from automated synthesis, further approaches might include the dynamic simulation of architectures, based on the outcome of the discussed measures.

For the further validation of the **procedural model**, discussions with practitioners are inevitable, which can in the future lead to a profound and practical approach for the management of product architectures, based on detailed workflow- and role-descriptions. The main tasks are described within the procedural model, yet their weighting and balanced application could not yet be based on empirical data. Additionally, a large amount of methods were analyzed to be integrated into the approach, yet many approaches, for example from the area of variant management, were not considered, and provide potential for further considerations.

The threefold approach for the management of product architectures provides a sound basis rooted in the MDM approach. The history and recent development of the product architecture, its implications and dependencies, suggest that considerable work lies ahead. In industry in particular, the awareness and need for systems architecting are vast, yet the transfer of scientific results and the validation of their practicability in industry requires a significant amount of effort in the future.

10. References

ADELI 1994

Adeli, H.: Advances in Design Optimization. London: Chapman & Hall 1994. ISBN: 0-412-53730-3.

AHLEMEYER & KÖNIGSWIESER 1998

Ahlemeyer, H. W.; Königswieser, R.: Komplexität managen: Strategien, Konzepte und Fallbeispiele. Wiesbaden: Gabler Verlag 1998. ISBN: 978-3409193160.

AKAO 1992

Akao, Y.: QFD: Quality Function Deployment. Wie die Japaner Kundenwünsche in Qualität umsetzen. Landsberg: Verlag Moderne Industrie: 1992.

ALBERT & BARABÁSI 2002

Albert, R.; Barabási, A.-L.: Statistical mechanics of complex networks. Reviews of Modern Physics 74 (2002) 1, pp. 47-97.

ANDERSON 2008

Anderson, C.: The Long Tail: Why the Future of Business is Selling Less of More. New York: Hyperion 2008. ISBN: 978-1401309664.

ANTONSSON & CAGAN 2001

Antonsson, E.K.; Cagan, J. (Eds.): Formal engineering design synthesis. New York: Cambridge University Press, 2001.

ARNOLD et al. 2005

Arnold, V.; Dettmering, H.; Engel, T.; Karcher, A.: Product Lifecycle Management beherrschen. Berlin: Springer 2005. ISBN: 978-3-540-22997-1.

Avak 2007

Avak, B.: Variant Management of Modular Product Families in the Market Phase. Dissertation, ETH Zürich, Zürich (2007).

BARABÁSI 2003

Barabási, A.-L.: Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life. New York: Plume 2003. ISBN: 0-452-28439-2.

BARTON & LOVE 2000

Barton, J. A.; Love, D. M.: Design decision chains as a basis for design analysis. In: Journal Engineering Design, 11(2000) 3, pp. 283-297.

BAUER & MEERKAMM 2007

Bauer, S.; Meerkamm, H.: Decision making with interdependent objectives in Design for X. In: Proceedings of the International Conference on Engineering Design, ICED’07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

BAUMBERGER 2007

Baumberger, G. C.: Methoden zur kundenspezifischen Produktdefinition bei individualisierten Produkten. Technische Universität München, München (2007).

BECK 2004

Beck, D.: Übersicht über Verfahren zum Umgang mit komplexen Aufgabenstellungen. In: Fisch, R. et al. (Eds.): Komplexitätsmanagement - Methoden zum Umgang mit komplexen Aufgabenstellungen in Wirtschaft, Regierung und Verwaltung. Wiesbaden: VS Verlag für Sozialwissenschaften 2004.

BERKOVICH et al. 2009

Berkovich, M.; Esch, S.; Leimeister, J. M.; Krcmar, H.: Requirements engineering for hybrid products as bundles of hardware, software and service elements – a literature review. In: Proceedings of the 9. Internationale Tagung der Wirtschaftsinformatik. Wien, Österreich 2009.

BERNARD 1999

Bernard, R.: Early Evaluation of Product Properties within the Integrated Product Development. Dissertation, Technische Universität München, Lehrstuhl für Produktentwicklung, München (1999).

BIGGS et al. 1999

Biggs, N. E.; Lloyd, K.; Wilson, R. J.: Graph Theory 1736-1936. New York: Oxford University Press 1999.

BLACKENFELT 2001

Blackenfelt, M.: Managing complexity by product modularisation: Balancing the aspects of technology and business during the design process. Stockholm: Department of Machine Design, Royal Institute of Technology 2001. ISBN: 1400-1179.

BLESSING & CHAKRABARTI 2009

Blessing, L. T. M.; Chakrabarti, A.: DRM, a Design Research Methodology. London: Springer 2009. ISBN: 978-1-84882-586-4.

BOARDMAN & SAUSER 2008

Boardman, J.; Sauser, B.: Systems Thinking: Coping with 21th Century Problems. Boca Raton: CRC Press 2008. ISBN: 978-1-4200-5491-0.

BOEHM 1988

Boehm, B. W.: A Spiral Model of Software Development and Enhancement. Computer 21 (1988) 5, pp. 61-72.

BONGULIELMI 2003

Bongulielmi, L.: Die Konfigurations- & Verträglichkeitsmatrix als Beitrag zur Darstellung konfigurationsrelevanter Aspekte im Produktentstehungsprozess. Dissertation, ETH Zürich, Zürich (2003).

BONGULIELMI et al. 2002

Bongulielmi, L.; Henseler, P.; Puls, C.; Meier, M.: The K- & V-Matrix-Method in Comparison with Matrix-Based Methods supporting Modular Product Family Architectures. In: Proceedings of NordDesign 2002 - Visions and Values in Engineering Design, 14.-16.08.2002. Trondheim: Norwegian University of Science and Technology 2002

BONJOUR et al. 2009

Bonjour, E.; Deniaud, S.; Dulmet, M.; Harmel, G.: A Fuzzy Method for Propagating Functional Architecture Constraints to Physical Architecture. Journal of mechanical design 131 (2009) 6, pp. 061002.

BOOCH 1994

Booch, G.: Objektorientierte Analyse und Design. Bonn: Addison Wesley 1994.

BOOTHROYD et al. 2002

Boothroyd G., Dewhurst P., Knight W.: Product Design For Manufacture and Assembly. Marcel Dekker Inc., New York, 2002.

BOULDING 1956

Boulding, K.: General Systems Theory: The Skeleton of Science. Management Science 2 (1956) 3, pp. 197-208.

BRADLEY & YASSINE 2006

Bradley, J. A.; Yassine, A. A.: On the Use of Network Analysis in Product Development Teams. In: ASME International Design Engineering Technical Conferences and 18th International Conference on Design Theory and Methodology (DTM), Philadelphia, PA, 10.-13.09.2006. New York: ASME 2007, ISBN: 0-7918-4258-4.

BRAHA & BAR-YAM 2004

Braha, D.; Bar-Yam, Y.: Topology of large-scale engineering problem-solving networks. Physical Review E 69 (2004) 1, pp. 016113.

BRAND et al. 2004

Brand, F.; Schaller, F.; Völker, H.: Transdisziplinarität. Bestandsaufnahme und Perspektiven. Göttingen: Universitätsverlag Göttingen 2004. ISBN: 3-930457-37-7.

BRAUN & DEUBZER 2007

Braun, T.; Deubzer, F.: New variant management using multiple-domain mapping. In: Proceedings of the International Design Structure Matrix Conference DSM 2007. Munich: TUM, 2007.

BROWNING 1998

Browning, T. R.: Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development. Dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1998.

BROWNING 2001

Browning, T. R.: Applying the Design Structure Matrix to System Decomposition and Integration Problems: a Review and New Directions. IEEE Transactions on Engineering Management 48 (2001) 3, pp. 292-306.

BROWNING 2009

Browning, T. R.: The Many Views of a Process: Toward a Process Architecture Framework for Product Development Processes. Systems Engineering 12 (2009) 1, pp. 69-99.

BROWNING & RAMASESH 2007

Browning, T. R.; Ramasesh, R. V.: A Survey of Activity Network-Based Process Models for Managing Product Development Projects. Production and Operations Management 16 (2007) 2, pp. 217-240.

BULLINGER et al. 2003

Bullinger, H.-J.; Kiss-Preußinger, E.; Spath, D.: Automobilentwicklung in Deutschland - Wie sicher ist die Zukunft? - Studie - Chancen, Potentiale und Handlungsempfehlungen für 30 Prozent mehr Effizienz. Stuttgart: Fraunhofer IRB Verlag 2003. ISBN: 3-8167-6388-X.

CAGAN 2001

Cagan, J.: Engineering Shape Grammars. In: Antonsson, E. K.; Cagan, J. (Eds.): Formal engineering design synthesis. New York: Cambridge University Press, 2001.

CAGAN et al. 2005

Cagan, J.; Campbell, M.; Finger, S.; Tomiyama, T.: A Framework for Computational Design Synthesis: Model and Applications. Journal of Computing and Information Science in Engineering 5 (2005) 3, pp. 171-181.

CAMELO et al. 2007

Camelo, D.; Mulet, E.; Vidal, R.: Function and Behavior Representation for Supporting Flexible Exploration and Generation in a Functional Model for Conceptual Design. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

CAMI & DEO 2008

Cami, A.; Deo, N.: Techniques for Analyzing Dynamic Random Graph Models of Web-Like Networks: An Overview. Wiley-Interscience 51 (2008) 4, pp. 211-255.

CAO et al. 2008

Cao, D.; Fu, M. W.; Gu, Y.; Jia, H.: Port-based Ontology Modeling for Conceptual Design. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE, 03.-06.08.2008, Brooklyn, NY, 2008.

CAVALLUCCI et al. 2002

Cavallucci, D.; Lutz, P.; Kucharavy, D.: Converging in Problem Formulation: A different Path in Design. In: Proceedings of the ASME Design Engineering Technical Conferences DETC/DTM 2002, Montreal, Canada, 29.09.-02.10.2002.

CHAKRABARTI 2002

Chakrabarti, A.: Engineering Design Synthesis: Understanding, Approaches and Tools. Berlin: Springer, 2002.

CHECKLAND 1993

Checkland, P. : Systems Thinking, Systems Practice. Chichester: John Wiley & Sons, Inc. 1993. ISBN: 0-471-27911-0.

CHEN 2005

Chen, S.-J.: An Integrated Methodological Framework for Project Task Coordination and Team Organization in Concurrent Engineering. Concurrent Engineering 13 (2005) 3, pp. 185-197.

CHENG 1997

Cheng, G.: Genetic Algorithms & Engineering Design. New York: John Wiley & Sons 1997. ISBN: 0-471-12741-8.

CHMARRA et al. 2008

Chmarra, M. K.; Arts, L.; Tomiyama, T.: Towards Adaptable Architecture. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE, 03.-06.08.2008, Brooklyn, NY, 2008.

CHURCHMAN et al. 1961

Churchman, C. W.; Ackoff, R. L.; Arnoff, E. L.: Operations Research: Eine Einführung in die Unternehmensforschung. Wien: R. Oldenbourg 1961.

CLARK & FUJIMOTO 1991

Clark, K. B.; Fujimoto, T.: Product Development Performance: Strategy, Organization, and Management in the World Auto Industry. Boston: Havard Business School Press 1991. ISBN: 0-87584-245-3.

CLARKSON & HAMILTON 2000

Clarkson, P. J.; Hamilton, J. R.: Signposting', A Parameter-Driven Task-Based Model of the Design Process. Research in Engineering Design 12 (2000) 1, pp. 18-38.

CLARKSON et al. 2001

Clarkson, P. J.; Simons, C.; Eckert, C.: Predicting Change Propagation in Complex Design. In: Proceedings of the ASME 2001 Design Engineering Technical Conferences DETC 2001, Pittsburgh, 09.-12.09.2001, Pittsburgh, PA, 2001.

CLARKSON et al. 2004

Clarkson, P. J.; Simons, C. S.; Eckert, C.: Predicting Change Propagation in Complex Design. Journal of Mechanical Design 126 (2004) 5, pp. 788-797.

COVALIU & OLIVER 1995

Covaliu, Z.; Oliver, R. M.: Representation and solution of decision problems using sequential decision diagrams. In: Management Science, 41 (1995) 12, pp. 1860–1881.

COAD & YOURDON 1994

Coad, P.; Yourdon, E.: Objektorientierte Analyse. München: Prentice Hall 1994.

COATANEA et al. 2008

Coatanea, E.; Alizon, F.; Christophe, F.; Yannou, B.: Selecting Technology Alternatives for Product Families Through Technological Coverage and Functional Verification. In: Proceedings of the ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE, 03.-06.08.2008, Brooklyn, NY, 2008.

COLLINS et al. 2008

Collins, S. T.; Yassine, A. A.; Borgatti, S. P. : Evaluating Product Development Systems Using Network Analysis. Systems Engineering (2008)

CRAWLEY et al. 2004

Crawley, E.; de Weck, O.; Eppinger, S. D.; Magee, C.; Moses, J.; Seering, W.; Schindall, J.; Wallace, D.; Whitney, D.: The Influence of Architecture in Engineering Systems, MIT Engineering Systems Symposium 2004. Boston, 2004.

DAENZER 1979

Daenzer, W. F.: Systems Engineering: Leitfaden zur methodischen Durchführung umfangreicher Planungsaufgaben. Köln: Peter Hanstein Verlag 1979. ISBN: 3-7756-6200-6.

DAENZER & HUBER 1999

Daenzer, W. F.; Huber, F. (Eds.): Systems Engineering. Methode und Praxis. Zürich: Verl. Industrielle Organisation, 1999.

D'AMELIO & TOMIYAMA 2007

D'Amelio, V.; Tomiyama, T.: Predicting the Unpredictable Problems in Mechatronics Design. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

DAMIC & MONTGOMERY 2003

Damic, V.; Montgomery, J.: Mechatronics by Bond Graphs: An Object-Oriented Approach to Modelling and Simulation. Berlin: Springer, 2003.

DANILOVIC 2006

Danilovic, M.: Bring your suppliers into your projects—Managing the design of work packages in product development. Journal of Purchasing & Supply Management 12 (2006) 5, pp. 246–257.

DANILOVIC & BÖRJESSON 2001

Danilovic, M., Börjesson, H.: Participatory Dependency Structure Matrix Approach. In: Proceedings of the 3rd Dependency Structure Matrix (DSM) International Workshop, Cambridge, MA, 29.-30.10.2001, Cambridge, MA,: Massachusetts Institute of Technology 2001.

DANILOVIC & BROWNING 2007

Danilovic, M., Browning, T. R.: Managing complex product development projects with design structure matrices and domain mapping matrices. International Journal of Project Management 25 (2007) 3, pp. 300–314.

DANILOVIC & SANDKULL 2004

Danilovic, M.; Sandkull, B.: The use of dependence structure matrix and domain mapping matrix in managing uncertainty in multiple project situations. International Journal of Project Management 23 (2004) 3, pp. 193-203.

DE BOER 1989

De Boer, S. J.: Decision Methods and Techniques in Methodical Engineering Design. De Lier: Academisch Boeken Centrum 1989. ISBN: 90-72015-32-0.

DE LIT & DELCHAMBRE 2003

De Lit, P.; Delchambre, A.: Integrated Design of a Product Family and Its Assembly System. Dordrecht: Kluwer Academic Publishers 2003. ISBN: 978-1402074370.

DE POEL 2007

De Poel, I. v.: Methodological problems in QFD and directions for future development. Research in Engineering Design 18 (2007) 1, pp. 21-36.

DE WECK 2007

De Weck, O. L.: On the Role of DSM in Designing Systems and Products for Changeability. In: Proceedings of the International Design Structure Matrix Conference DSM 2007, Munich, TUM, 2007.

DEUBZER et al. 2005

Deubzer, F.; Kreimeyer, M.; Junior, T.; Rock, B.: Der Änderungsmanagement Report 2005. CiDaD Working Paper Series 1 (2005) 1, pp. 1-12.

DEUBZER & LINDEMANN 2008

Deubzer, F.; Lindemann, U.: Functional Modeling for Design Synthesis using MDM Methodology. In: Proceedings of the International Design Structure Matrix Conference DSM 2008, Stockholm: KTH, 2008.

DEUBZER et al. 2008

Deubzer, F.; Braun, T.; Maurer, M.; Lindemann, U.: Applying the Multiple Domain Mapping Approach to Variant Management. In: Proceedings of the 11th International Design Conference DESIGN 2008, Dubrovnik, Croatia: The Design Society, 2008.

DEUBZER & LINDEMANN 2009a

Deubzer, F.; Lindemann, U.: MDM Application to Interrelate Hierarchical Layers of Abstraction. In: Proceedings of the International Design Structure Matrix Conference 2009, Charlotte, SC, Clemson University, 2009.

DEUBZER & LINDEMANN 2009b

Deubzer, F.; Lindemann, U.: Networked Product Modeling – Use And Interaction Of Product Models And Methods During Analysis And Synthesis. In: Proceedings of the International Conference on Engineering Design. Stanford University, CA: The Design Society, 2009.

DEUBZER & LINDEMANN 2009c

Deubzer, F.; Lindemann, U.: Product Architecture Definition and Analysis using Matrix-Based Multiple-Domain Approaches. In: Proceedings of the ASME IDETC/CIE 2009, 30.08.-02.09.2009, San Diego, CA, 2009.

DIAZ 1998

Diaz, C. A.: Product Re-Configurability and Product Introduction. In: Concurrent Engineering 6 (1998) 3, pp. 172-177.

DIEPOLD et al. 2009

Diepold, K. J.; Winkler, F. J.; Lohmann, B.; Kortler, S.: A Framework for DSM-Based Pre-Modelling Analysis of Complex Systems. In: Proceedings of the International Design Structure Matrix Conference 2009, Charlotte, SC, Clemson University, 2009.

DIESTEL 2006

Diestel, R.: Graph Theory. Berlin: Springer 2006. ISBN: 978-3-540-26183-4.

DOD 1997

U.S. Department of Defense, C4ISR Architecture Working Group: C4ISR Architecture Framework, Version 2.0. Washington, DC: U.S. Department of Defense 1997.

DOMSCHKE & DREXL 2002

Domschke, W.; Drexl, A.: Einführung in Operations Research. Berlin: Springer 2002. ISBN: 3-540-42950-6.

DONG 1995

Dong, J.: Organization Structures, Concurrent Engineering, and Computerized Enterprise Integration. Concurrent Engineering: Research and Applications 3 (1995) 3, pp. 167-176.

DÖRNER 1992

Dörner, D.: Die Logik des Misslingens. Hamburg: Rowohlt 1992. ISBN: 3-499-19314-0.

DORST & VERMAAS 2005

John Gero's Function-Behaviour-Structure Model of Designing: A Critical Analysis. In: Research in Engineering Design, 16 (2005) 1-2, pp. 17-26.

DOVE 2006

Dove, R.: Engineering Agile Systems: Creative-Guidance Frameworks for Requirements and Design. In: Proceedings of the 4th Annual Conference on Systems Engineering Research (CSER), 07.-08.04.2006, Los Angeles, CA, 2006

DU et al. 2001

Du, X.; Jiao, J.; Tseng, M.: Architecture of Product Family: Fundamentals and Methodology. In: Concurrent Engineering 9 (2001) 4, pp. 309-325.

DU et al. 2002

Du, X.; Jiao, J.; Tseng, M.: Graph Grammar Based Product Family Modeling. In: Concurrent Engineering 10 (2002) 2, pp. 113-128.

DUHOVNIK et al. 2006

Duhovnik, J.; Kusar, J.; Tomazevic, R.; Starbek, M.: Development Process with Regard to Customer Requirements. Concurrent Engineering, 14 (2006) 3, pp. 67-82.

EBEN et al. 2008

Eben, K.; Biedermann, W.; Lindemann, U.: Modeling Structural Change over Time: Requirements and first Methods. In: Proceedings of the International Design Structure Matrix Conference DSM 2008, Stockholm: KTH, 2008.

ECKERT et al. 2004

Eckert, C.; Clarkson, P. J.; Zanker, W.: Change and Customisation in Complex Engineering Domains. In: Research in Engineering Design, 15 (2004) 1, pp. 1-21.

EHRLENSPIEL 2009

Ehrlenspiel, K.: Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit. München: Carl Hanser Verlag 2009. ISBN: 978-3-446-42013-7.

EHRLENSPIEL et al. 2007

Ehrlenspiel, K.; Kiewert, A.; Lindemann, U.; Hundal, M. S.: Cost-Efficient Design. Berlin: Springer 2007. ISBN: 3540346473.

ENGEL & BROWNING 2008

Engel, A.; Browning, T. R.: Designing Systems for Adaptability by Means of Architecture Options. Systems Engineerig 11 (2008) 2, pp. 125-146.

EPPINGER 2001

Eppinger, S. D.: Patterns of product development Interactions. In: Proceedings of the International Conference on Engineering Design, ICED'01, Glasgow, UK, 283-290. Bury St Edmunds: Professional Engineering ISBN: 1-8605-8354-7.

EPPINGER et al. 1997

Eppinger, S. D.; Nukala, M. V.; Whitney, D. E.: Generalised Models of Design Iteration Using Signal Flow Graphs. Research in Engineering Design 9 (1997) 2, pp. 112-123.

EPPINGER & SALMINEN 2001

Eppinger, S. D.; Salminen, V.: Patterns of Product Development Interactions. In: Culley, S. (Ed.): 13th International Conference on Engineering Design, ICED 01, Glasgow, 21.-23.08.2001. Bury St. Edmunds: Professional Engineering Publ. 2001, pp. 283-290. ISBN: 1-86058-354-7.

ERDEN et al. 2008

Erden, M. S.; Komoto, H.; Van Beek, T. J.; D'Amelio, V.; Echavarria, E.; Tomiyama, T.: A review of function modeling: Approaches and applications, 22 (2008) 2, pp. 147-169.

FARRELL & SIMPSON 2008

Farrell, R. S.; Simpson, T. W.: A Method to Improve Platform Leveraging in a Market Segmentation Grid for an Existing Product Line. In: Journal of Mechanical Design 130 (2008) 3

FELGEN 2007

Felgen, L.: Systemorientierte Qualitätssicherung für mechatronische Produkte. Technische Universität München, München (2007).

FELGEN et al. 2005a

Felgen, L.; Deubzer, F.; Lindemann, U.: Vorgehensmodell zur Identifikation kritischer Merkmale von mechatronischen Systemen. In: Proceedings of the Mechatronik 2005 - Innovative Produktentwicklung, Wiesloch, 01.-02.06. 2005, pp. 253-272, Düsseldorf: VDI-Verlag, 2005.

FELGEN et al. 2005b

Felgen, L.; Deubzer, F.; Lindemann, U.: Complexity management during the analysis of mechatronic systems. In: Proceedings of the International Conference on Engineering Design ICED'05, Melbourne (Australia), Institution of Engineers Australia, 2005.

FIGEL 1988

Figel, K.: Optimieren beim Konstruieren: Einsatz von Optimierungsverfahren, CAD und Expertensysteme. München: Carl Hanser Verlag 1988. ISBN: 3-446-15344-6.

FISCH & BECK 2004

Fisch, R.; Beck, D.: Komplexitätsmanagement: Methoden zum Umgang mit komplexen Aufgabenstellungen in Wirtschaft, Regierung und Verwaltung. Wiesbaden: VS Verlag für Sozialwissenschaften 2004. ISBN: 3-531-14437-5.

FIXSON 2007

Fixson, S. K.: Modularity and Commonality Research: Past Developments and Future Opportunities. Concurrent Engineering 15 (2007) 2, pp. 85-111.

FOTSO et al. 2007

Fotso,B. M.; Dulmet, M.; Bonjour,E.: Design of product families based on a modular architecture. In: Proceedings of the First International Conference on Multidisciplinary Design Optimization and Applications, 28.11.2007, Besançon, France, 2007.

FRICKE & SCHULZ 2005

Fricke, E.; Schulz, A. p. : Design for Changeability (DfC): Principles to Enable Changes in Systems Throughout their Entire Lifecycle. Systems Engineering 8 (2005) 4, pp. 342-358.

FRIEDENTHAL et al. 2009

Friedenthal, S.; Moore, A.; Steiner, R.: A Practical Guide to Sysml: The Systems Modeling Language. Waltham, MA: Morgan Kaufmann, 2009

FUCHS 2004

Fuchs, D. K.: Prinzipien für die Problemanalyse in der Produktentwicklung. Technische Universität München, München (2004).

FUKUZAWA 2008

Fukuzawa, M.: A Generation-Selection Process of Product Architecture: A Case of Development of the Firmware in Digital MFP. Annals of Business Administrative Science ABAS 7 (2008) 1, pp. 1-18.

GAHR 2006

Pfadkostenrechnung individualisierter Produkte. Dissertation, Technische Universität München, Lehrstuhl für Produktentwicklung, München (2006).

GANESAN & PREVOSTINI 2006

Ganesan, S.; Prevostini, M.: Bridging the Gap between SysML and Design Space Exploration. In: Proceedings of the Forum on Specification & Design Languages FDL 2006, Darmstadt, Germany, 19.-22.09.2006. ECSI, 2006.

GAUSEMEIER et al. 1996

Gausemeier, J., Fink, A., Schlake, O.: Szenario-Management: Planen und Führen mit Szenarien. München: Carl Hanser 1996.

GAUSEMEIER et al. 2001

Gausemeier, J.; Ebbesmayer, P. ; Kallmeyer, F.: Produktinnovation. Strategische Planung und Entwicklung der Produkte von morgen. München: Carl Hanser 2001.

GAUSEMEIER et al. 2006

Gausemeier, J.; Hahn, A.; Kespohl, H. D.; Seifert, L.: Vernetzte Produktentwicklung - Der erfolgreiche Weg zum Global Engineering Network. München: Carl Hanser 2006. ISBN: 3-446-22725-3.

GERO 1990

Gero, J. S.: Design Prototypes: A Knowledge Representation Schema for Design. In: AI Magazine 11 (1990) 4, pp. 27-36.

GERO & KANNENGIESSER 2004

Gero, J. S.; Kannengiesser, U.: The Situated Function-Behaviour-Structure Framework. In: Design studies, 25 (2004) 4, pp. 373-391.

GERSHENSON et al. 2004

Gershenson, J. K.; Prasad, G. J.; Zhang, Y.: Product modularity: measures and design methods. In: Journal of Engineering Design 15 (2004) 1, pp. 33-51.

GILMORE & PINE 2000

Gilmore, J. H.; Pine, B. J.: Markets of One: Creating Customer-Unique Value through Mass Customization. Boston: Havard Business School Press 2000. ISBN: 978-1578512386.

GÖPFERT 1998

Göpfert, J.: Modulare Produktentwicklung. Zur gemeinsamen Gestaltung von Technik und Organisation. Wiesbaden: Gabler 1998.

GORBEA et al. 2008

Gorbea, C.; Spielmannleitner, T.; Lindemann, U.; Fricke, E.: Analysis of Hybrid Vehicle Architectures Using Multiple Domain Matrices. In: Proceedings of the International Design Structure Matrix Conference DSM 2008, Stockholm: KTH, 2008.

GRABNER & NOTHHAFT 2002

Grabner, J.; Nothhaft, R.: Konstruieren von PKW-Karosserien: Grundlagen, Elemente und Baugruppen, Vorschriftenübersicht. Berlin: Springer 2002. ISBN: 3-540-43290-6.

GRAEBSCH et al. 2007

Graebsch, M.; Lindemann, U.; Weiss, S.: Lean Development in Deutschland. München: Dr. Hut 2007. ISBN: 978-3-89963-496-9.

GRAEBSCH et al. 2009

Graebsch, M.; Deubzer, F.; Lindemann, U.: Graph Representation of Physical Effects Networks in Conceptual Design. In: Proceedings of the International Conference on Engineering Design. Stanford University, CA: The Design Society, 2009.

GUENOV & BARKER 2005

Guenov, M. D.; Barker, S. G.: Application of Axiomatic Design and Design Structure Matrix to the Decomposition of Engineering Systems. In: Journal Systems Engineering 8 (2005) 1, pp. 29-40.

GULATI & EPPINGER 1996

Gulati, R. K.; Eppinger, S. D.: The Coupling of Product Architecture and Organizational Structure Decision. MIT Sloan School of Management Working Paper 3906 (1996)

HAN & KAMBER 2001

Han, J.; Kamber, M.: Data Mining: Concepts and Techniques. San Francisco: Morgan Kaufmann, 2001.

HANDKE 2000

Handke, S.: Konzept zur Strukturplanung komplexer technischer Systeme: Systematische Produktstrukturierung zur Vereinfachung technischer und organisatorischer Schnittstellen im Produktentstehungsprozess. Heimsheim: Jost Jetter Verlag 2000. ISBN: 3-931388-38-7.

HATAMURA 2006

Hatamura, Y.: Decision-Making in Engineering Design: Theory and Practice. London: Springer 2006. ISBN: 978-1-84628-000-9.

HAUSE 2006

Hause, M.: The SysML Modelling Language. In: Proceedings of the Fifth European Systems Engineering Conference EuSEC, Edinburgh, Scotland, UK, 18.-20.09.2006. INCOSE, 2006.

HELMS & SHEA 2010

Helms, B.; Shea, K.: Object-oriented concepts for computational synthesis. In: Proceedings of the 12th International Design Conference DESIGN 2010, Dubrovnik, Croatia: The Design Society, 2010.

HELLENBRAND & LINDEMANN 2008

Hellenbrand, D.; Lindemann, U.: Using the DSM to Support the Selection of Product Concepts. In: Proceedings of the International Design Structure Matrix Conference DSM 2008, Stockholm: KTH, 2008.

HEMEL & KELDMANN 1996

Hemel van, C. G.; Keldmann, T.: Applying "Design for X" Experience in Design for Environment. In: Huang, G. Q. (Ed.): Design for X: concurrent engineering imperatives. London: Chapman & Hall 1996, pp. 72-95. ISBN: 978-0412787504.

HENDERSON & CLARK 1990

Henderson, R. M.; Clark, K. B.: Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. Administrative Science Quarterly (Special Issue: Technology, Organizations, and Innovation.) 35 (1990) 1, pp. 9-30.

HEPPERLE et al. 2009a

Hepperle, C.; Thanner, S.; Mörtl, M.; Lindemann, U.: An integrated product lifecycle model and interrelations in-between the lifecycle phases. In: Proceedings of the International Conference on Product Lifecycle Management, PLM'09, Bath, UK, 6.-8.7.2009. Garching, Germany: Inderscience Enterprises Ltd.

HEPPERLE et al. 2009b

Hepperle, C.; Deubzer, F.; Wiedemann, P.; Lindemann, U.; Liebl, J.; Hallmannsegger, M.; Hübner, W.: Customer perception of vehicle dynamics and its transfer to technical characteristics. In: Proceedings of the International Conference on Engineering Design. Stanford University, CA: The Design Society, 2009.

HERFELD 2007

Herfeld, U.: Matrix-basierte Verknüpfung von Komponenten und Funktionen zur Integration von Konstruktion und numerischer Simulation. Technische Universität München, München (2007).

HOISL et al. 2008

Hoisl, F.; Shea, K.; Helms, B.: Towards Representing, Evolving and Networking Engineering Knowledge for Computational Design Synthesis. In: Proceedings of the 11th International Design Conference DESIGN 2008, Dubrovnik, Croatia: The Design Society, 2008.

HOLT & PERRY 2008

Holt, J.; Perry, S.: SysML for Systems Engineering. Stevenage, UK: Institution of Engineering and Technology, 2008.

HONOUR & BROWNING 2007

Honour, E. C.; Browning, T. R.: Dynamic Optimization of Systems of Systems Using Value Measurement. Journal of Integrated Design & Process Science 11 (2007) 2, pp. 33-53.

HUANG 1996

Huang, G. Q.: Design for X. Concurrent Engineering Imperatives. 1st Edition Aufl. London: Chapman & Hall 1996.

HUBKA & EDER 1988

Hubka, V.; Eder, W. E.: Theory of Technical Systems: A Total Concept Theory for Engineering Design. Berlin: Springer 1988. ISBN: 978-0387174518.

HUBKA & EDER 1996

Hubka, V.; Eder, W. E.: Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge. London: Springer 1996. ISBN: 3-540-19997-7.

ILIE et al. 2008

Ilie, D.; Fischer, F.; Lindemann, U.: Analysis of the Information Environment in the Context of Target and Requirements Management in the Automotive Industry. In: Proceedings of the ASME 2008 International Design Engineering Technical

Conferences and Computers and Information in Engineering Conference 2008, Brooklyn, 03.-06.08.2008, Brooklyn, NY, 2008.

JACKSON 1999

Jackson, P.: Introduction to Expert Systems. Bonn: Addison Wesley, 1999.

JIAO & TSENG 1999

Jiao, J.; Tseng, M.: A methodology of developing product family architecture for mass customization. In: Journal of Intelligent Manufacturing 10 (1999) 1, pp. 3-20.

JIAO & CHEN 2006

Jiao, J.; Chen, C.-H.: Customer Requirement Management in Product Development: A Review of Research Issues. Concurrent Engineering 14 (2006) 3, pp. 173-185.

JIAO et al. 2006

Jiao, J.; Simpson, T. W.; Siddique, Z.: Product Family Design and Platform-Based Product Development: A State-of -the-Art Review. Journal of Intelligent Manufacturing 18 (2006) 1, pp. 5-29.

JOHNSON et al. 2007

Johnson, T. A.; Paredis, C. J. J.; Jobe, J. M.; Burkhart, R.: Modeling Continuous System Dynamics in SysML. In: Proceedings of the ASME International Mechanical Engineering Congress and Exposition IMECE 2007, Seattle, Washington, 11.-15.11.2007.

JUNG 2006

Jung, C.: Anforderungsklärung in interdisziplinärer Entwicklungsumgebung. Dissertation, Technische Universität München, Lehrstuhl für Produktentwicklung, München (2006).

KAIN et al. 2009

Kain, A.; Kirschner, R.; Goldt, M.; Lindemann, U.; Gunkel, J.; Klendauer, R.; Schneider, M.; Wastian, M.: A method to identify relevant stakeholders to be integrated in new product development processes. In: Proceedings of the International Conference on Research into Design, ICoRD '09. Bangalore, India, 2009.

KLEIN MEYER et al. 2007

Klein Meyer, J. S.; Cabannes, G.; Lafon, P.; Troussier, N.; Roucoules, L.; Gidel, T.: Product Modelling for Design Alternatives Using Optimisation and Robustness Analysis. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

KNOBLICH 1997

Knoblich, G.: Repräsentationswechsel als Grundlage von Einsicht. Experimentalpsychologische Untersuchung in der Domäne der Streichholzalgebra. Hamburg: Universität Hamburg 1997. ISBN: 0943-5204.

KOH et al. 2009

Koh, E. C. Y.; Keller, R.; Eckert, C. M.; Clarkson, P. J.: Change Propagation Modelling to Support the Selection of Solutions in Incremental Change. In: Proceedings of the International Conference on Research into Design, ICoRD '09. Bangalore, India, 2009.

KORNMEIER & RUDOLPH 2006

Kornmeier, T.; Rudolph, S.: Topological Synthesis of Shell Structures. In: Proceedings of the ASME 2006 Design Engineering Technical Conferences DETC 2006, Pittsburgh, 10.-13.09.2006, Philadelphia, PA, 2006.

KOSSIAKOFF & SWEET 2003

Kossiakoff, A.; Sweet, W. N.: Systems Engineering: Principles and Practice. Hoboken: John Wiley & Sons, Inc. 2003. ISBN: 0-471-23443-5.

KREIMEYER 2010

Kreimeyer, M.: A Structural Measurement System for Engineering Design Processes. Dissertation, Technische Universität München, Lehrstuhl für Produktentwicklung, München (2010).

KREIMEYER et al. 2006

Kreimeyer, M.; Herfeld, U.; Deuber, F.; Lindemann, U.: Effiziente Zusammenarbeit von Konstruktions- und Simulationsabteilungen in der Automobilindustrie. CiDaD Working Paper Series 2 (2006) 1, pp. 1-13.

KRISHNAN & ULRICH 2001

Krishnan, V.; Ulrich, K. T.: Product Development Decisions: A Review of the Literature. Management Science 47 (2001) 1, pp. 1-21.

KRUCHTEN 1995

Kruchten, P.: Architectural Blueprints - The “4+1” View Model of Software Architecture. In: IEEE Software 12 (1995) 6, pp. 42-50.

KUMAR & ALLADA 2005

Kumar, R.; Allada, V.: Customer Need Driven Function-Behavior Platform Formation. In: Proceedings of the IDETC/CIE 2005 ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Long Beach, 24.-28.09.2005. Long Beach: ASME, 2005.

KUNZ et al. 1996

Kunz, J. C.; Luiten, G. T.; Fischer, M. A.; Jin, Y.; Levitt, R. E.: CE4: of Product, Process, Facility, and Organization. Concurrent Engineering: Research and Applications 4 (1996) 2, pp. 187-198.

KURFMAN et al. 2001

Kurfman, M. A.; Stone, R. B.; Rajan, J. R.; Wood, K. L.: Functional Modeling Experimental Studies. In: Proceedings of the ASME 2001 Design Engineering Technical Conferences DETC 2001, Pittsburgh, 09.-12.09.2001, Pittsburgh, PA, 2001.

KURTOGLU & CAMPBELL 2006

Kurtoglu, T.; Campbell, M. I.: A Graph Grammar based Framework for Automated Concept Generation. In: Proceedings of the 9th International Design Conference DESIGN 2006, Dubrovnik, Croatia: The Design Society, 2006.

KUSIAK 1999

Kusiak, A.: Engineering Design: Products, Processes, and Systems. San Diego: Academic Press 1999. ISBN: 0-12-430145-2.

KUSIAK 2002

Kusiak, A.: Integrated product and process design: a modularity perspective. In: Journal of Engineering Design 13 (2002) 3, pp. 223-231.

LASZLO 1995

Laszlo, E.: The Interconnected Universe: Conceptual Foundations of Transdisciplinary Unified Theory. Singapore: World Science 1995. ISBN: 981-02-2202-5.

LAUER et al. 2008

Lauer, W.; Felgen, L.; Ponn, J.; Hübner, W.; Lindemann, U.: Support of the Development Process by a new Approach using Multiple Views on Physical Effects. In: Proceedings of the FISITA World Automotive Congress, 14.-19-09-2008, Munich, 2008.

LAWSON & KARANDIKAR 1994

Lawson, M.; Karandikar, H. M.: A Survey of Concurrent Engineering. Concurrent Engineering: Research and Applications 2 (1994) 1, pp. 1-6.

LENZ & COCHRAN 2000

Lenz, R. K.; Cochran, D. S.: The Application of Axiomatic Design to the Design of the Product Development Organization. In: Proceedings of the 1st International Conference on Axiomatic Design ICAD2000, Cambridge, MA, 21.-23.06.2000, Cambridge: MIT, 2000.

LEVARDY & BROWNING 2009

Levardy, V.; Browning, T. R.: An Adaptive Process Model to Support Product Development Project Management. IEEE Transactions on Engineering Management 56 (2009) 4, pp. 600-620.

LEVIS 1999

Levis, A.: System Architectures. In: Sage, A. and Rouse W. B. (Eds.): Handbook of Systems Engineering and Management. New York: John Wiley & Sons, pp. 427-454.

LIEBL 2006

Liebl, J.: Energiemanagement – Ein Schlüssel für Effiziente Dynamik. In: VDI-Berichte Nr. 1975, VDI-Tagung Innovative Fahrzeugantriebe, Dresden, Germany, 09.-10.11.2006. Düsseldorf: VDI-Verlag 2006, S. 449-463. ISBN: 3-18-091975-2.

LIN 1999

Lin, Y.: General systems theory: a mathematical approach. New York: Kluwer Academic / Plenum Publishers 1999. ISBN: 0-306-45944-2.

LINDEMANN et al. 2006

Lindemann, U.; Reichwald, R.; Zäh, M. F.: Individualisierte Produkte - Komplexität beherrschen in Entwicklung und Produktion. Berlin: Springer 2006.

LINDEMANN 2007

Lindemann, U.: A Vision to Overcome "Chaotic "Design for X Processes in Early Phases. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

LINDEMANN 2009

Lindemann, U.: Methodische Entwicklung technischer Produkte: Methoden flexibel und situationsgerecht anwenden. Berlin: Springer 2009. ISBN: 978-3-642-01422-2.

LINDEMANN et al. 2009

Lindemann, U.; Maurer, M.; Braun, T.: Structural Complexity Management. Berlin: Springer 2009.

LINDQUIST et al. 2008

Lindquist, A.; Berglund, F.; Johannesson, H.: Supplier Integration and Communication Strategies in Collaborative Platform Development. Concurrent Engineering: Research and Applications 16 (2008) 1, pp. 23-35.

LIU et al. 2001

Liu, X. F.; Noguchi, K.; Zhou, W.: Requirement Acquisition, Analysis, and Synthesis in Quality Function Deployment. Concurrent Engineering: Research and Applications 9 (2001) 1, pp. 24-36.

LÖSCH 2001

Lösch, J.: Controlling der Variantenvielfalt - Eine koordinationsorientierte Konzeption zur Steuerung von Produktvarianten. Aachen: Shaker Verlag GmbH 2001. ISBN: 3-8265-9272-7.

LUH et al. 2011

Luh, D.-B-; Ko, Y.-T.; Ma, C.-H.: A structural matrix-based modelling for designing product variety. In: Journal of Engineering Design, 22 (2011) 1, pp. 1-29.

MAHER 1990

Maher, M. L: Process Models for Design Synthesis. AI Magazine 11 (1990) 4, pp. 49-58.

MAIER & FADEL 2001

Maier, J. R. A.; G. M. Fadel: Affordance: The Fundamental Concept in Engineering Design. In: Proceedings of the ASME 2001 Design Engineering Technical Conferences DETC 2001, Pittsburgh, 09.-12.09.2001, Pittsburgh, PA, 2001.

MAIER & FADEL 2006

Maier, J. R. A.; G. M. Fadel: Affordance Based Design: Status and Promise. In: Proceedings of the International Design Research Symposium, Seoul, South Korea, pp.67-80, 2006.

MAIER & RECHTIN 2000

Maier, M. W.; Rechtin, E.: The Art of Systems Architecting. 2 Aufl. Boca Raton: CRC Press 2000.

MALETZ 2008

Maletz, M.: Integrated Requirement Modeling: A Contribution towards the Integration of Requirements into a Holistic Product Lifecycle Management Strategy. Kaiserslautern: Technische Universität Kaiserslautern 2008. ISBN: 978-3-939432.

MALIK 2008

Malik, F.: Strategie des Managements komplexer Systeme. Ein Beitrag zur Management- Kybernetik evolutionärer Systeme. Bern: Paul Haupt Verlag 2008. ISBN: 978-325-07396-5.

MANN 2002

Mann, D.: Axiomatic Design and TRIZ: Compatibilities and Contradictions. In: Proceedings of the 2nd International Conference on Axiomatic Design ICAD2002, Cambridge, MA, 10.-11.06.2002, Cambridge: MIT, 2002.

MARTIN & ISHII 2002

Martin, M. V.; Ishii, K.: Design for variety: developing standardized and modularized product platform architectures. In: Research in Engineering Design 13 (2002) 4, pp. 213-235.

MATTHES 2011

Matthes, D.: Enterprise Architecture Frameworks Kompendium. Berlin: Springer, 2011.

MAURER 2007

Maurer, M. S.: Structural Awareness in Complex Product Design. Dissertation, Technische Universität München, München (2007).

MAURER & KESPER 2011

Maurer, M.; Kesper, H.: eFMEA – Raising Efficiency of FMEA by Matrix-Based Function and Failure Networks. In: In: Proceedings of the International Conference on Research into Design, ICoRD '11. Bangalore, India, 2011.

MCDERMOTT et al. 2009

McDermott, R. E.; Mikulak, R. J.; Beauregard, M. R.: The Basics of FMEA. Boca Raton: CRC Press 2009.

MCKENNA 2000

McKenna, R.: Marketing in an Age of Diversity. In: Gilmore, J. H. et al. (Eds.): Markets of One. Boston: Harvard Business School Press 2000, pp. 17-52.

MEYER 1997

Meyer, M. H.: Revitalize Your Product Lines Through Continuous Platform Renewal. In: Research in Technology Management 40 (1997) 2, pp. 17-28.

MIßLER-BEHR 1993

Mißler-Behr, M.: Methoden der Szenarioanalyse. Wiesbaden: Deutscher Universitätsverlag 1993.

MOD 2005

Ministry of Defense: MoD Architectural Framework Technical Handbook 1.0. London: UK Ministry of Defense 2005.

NEO & GUPTA 2003

Neo, N.; Gupta, P. : Graph-Theoretic Analysis of the World Wide Web: New Directions and Challenges. Matemática Contemporânea 25 (2003) pp. 49-69.

NEWMAN 2003

Newman, M. E. J.: The Structure and Function of Complex Networks. SIAM Review 45 (2003) 2, pp. 167-256.

NOVAK & EPPINGER 2001

Novak, S.; Eppinger, S. D.: Sourcing By Design: Product Complexity and the Supply Chain. Management Science 47 (2001) 1, pp. 189-204.

ÖLVANDER et al. 2009

Ölvander, J.; Lundén, B.; Gavel, H.: A computerized optimization framework for th morphological matrix applied to aircraft conceptual design. Computer-Aided Design 41 (2009) 3, pp. 187-196.

OH et al. 2007

Oh, S.; Park, B.; Park, S.; Hong, Y. S.: Design of Change-Absorbing System Architecture for the Design of Robust Products and Services. In: Stephanidis, C. (Ed.): Proceedings of the 12th international conference on Human-computer interaction HCI'07: applications and services, Beijing, 19.-21.06.2007, Beijing, China 2007.

O'NEAL 2003

O'Neal, J. S.: Analyzing the Impact of Changing Software Requirements: A Traceability-based Methodology. Dissertation, Louisiana State University, Department of Computer Science, Eunice, LA (2003).

OOKUBO et al. 2007

Ookubo, M.; Koji, Y.; Sasajiama, M.; Kitamura, Y.; Mizoguchi, R.: Towards Interoperability between Functional Taxonomies using an Ontology-based mapping. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

PAHL et al. 2007

Pahl, G.; Beitz, W.; Feldhusen, J.; Grote, K. H.; Wallace, K.; Blessing, L.: Engineering Design: A Systematic Approach. 3 Aufl. London: Springer 2007.

PARK & SIMPSON 2008

Park, J.; Simpson, T. W.: Toward an activity-based costing system for product families and product platforms in the early stages of development. In: International Journal of Production Research 46 (2008) 1, pp. 99-300.

PILLER 2001

Piller, F. T.: Mass Customization. Ein wettbewerbsstrategisches Konzept im Informationszeitalter. 2 Aufl. Wiesbaden: Gabler 2001.

PILLER et al. 2004

Piller, F. T.; Moeslein, K.; Stotko, C. M.: Does mass customization pay? An Economic Approach to Evaluate Customer Integration. Production Planning & Control 15 (2004) 4, pp. 435-444.

PILLER & STOTKO 2003

Piller, F. T.; Stotko, C. M.: Mass Customization und Kundenintegration: Neue Wege zum innovativen Produkt. Düsseldorf: Symposion Publishing 2003. ISBN: 3-936608-05-9.

PIMMLER & EPPINGER 1994

Pimmler, T. U.; Eppinger, S. D.: Integration Analysis of Product Decompositions. In: Hight, T. K. et al. (Eds.): ASME Design Technical Conferences, 6th International Conference on Design Theory and Methodology, DTM '94, Minneapolis, 11.-14.09.1994. New York: ASME 1994, ISBN: 0-7918-1282-0.

PINE 1993

Pine, B. J.: Mass Customization: The New Frontier in Business Competition. Boston: Havard Business School Press 1993. ISBN: 0-87584-372-7.

PONN 2007

Ponn, J.: Situative Unterstützung der methodischen Konzeptentwicklung technischer Produkte. Dissertation, Technische Universität München, München (2007).

PONN & LINDEMANN 2008

Ponn, J.; Lindemann, U.: Konzeptentwicklung und Gestaltung technischer Produkte: Optimierte Produkte - systematisch von Anforderungen zu Konzepten. Berlin: Springer 2008. ISBN: 978-3-540-68562-3.

PULM 2004

Pulm, U.: Eine systemtheoretische Betrachtung der Produktentwicklung. Dissertation, Technische Universität München, München (2004).

QIAN & GERO 1996

Qian, L., Gero, J. S.: Function-behavior-structure paths and their role in analogy-based design. In: Artificial Intelligence for Engineering, Design, Analysis and Manufacturing, 10 (1996) 4, pp. 289-312.

RAPP 1999

Rapp, T.: Produktstrukturierung. St. Gallen: Gabler Verlag 1999. ISBN: 978-3824470105.

RENNER 2007

Renner, I.: Methodische Unterstützung funktionsorientierter Baukastenentwicklung am Beispiel Automobil. Dissertation, Technische Universität München, München (2007).

RITCHEY 1998

Ritchey, T.: General Morphological Analysis:A general method for non-quantified modelling. In: Proceedings of the 16th EURO Conference on Operational Analysis 1998, Brussels, 1998.

RITCHEY 2006

Ritchey, T.: Problem Structuring using Computer-Aided Morphological Analysis. In: Journal of the Operational Research Society, Special Issue on Problem Structuring Methods (2006) 57, pp. 792-801.

ROOZENBURG & EEKELS 1990

Roozenburg, N.; Eekels, J.: Evaluation and Decision in Design - Bewerten und Entscheiden beim Konstruieren. Zürich: Heurista 1990. ISBN: 3-85693-0213.

ROOZENBURG & EEKELS 1995

Roozenburg, N.; Eekels, J.: Product Design: Fundamentals and Methods. Chichester: Wiley, 1995.

ROPOHL 1975

Ropohl, G.: Systemtechnik - Grundlagen und Anwendungen. München: Carl Hanser Verlag 1975. ISBN: 3-446-11829-2.

ROTH 2001

Roth, K.: Konstruieren mit Konstruktionskatalogen: Band II - Konstruktionskataloge. Berlin: Springer 2001. ISBN: 978-3-540-67026-1.

ROYCE 1987

Royce, W. W.: Managing the development of large software systems: concepts and techniques, Proceedings of the 9th international conference on Software Engineering. Monterey, California, United States, 1987.

SABISCH & TINTELNOT 1997

Sabisch, H.; Tintelnot, C.: Integriertes Benchmarking für Produkte und Produktentwicklungsprozesse. Berlin: Springer, 1997.

SADEK HASSANEIN 2008

Sadek Hassanein, T.: Ein modellorientierter Ansatz zur Konzeptentwicklung industrieller Produkt-Service Systeme. Dissertation, Ruhr-Universität Bochum, Bochum (2008).

SALHIEH & KAMRANI 1999

Salhieh, S. M.; Kamrani, A. K.: Macro level product development using design for modularity. In: Robotics and Computer-Integrated Manufacturing 15 (1999) 4, pp. 319-329.

SANDER 2001

Sander, S.: Konzept einer digitalen Lösungsbibliothek für die integrierte Produktentwicklung. Düsseldorf: VDI Verlag 2001. ISBN: 3-18-334201-4.

SANTE et al. 2007

Sante van, T.; Kemmeren, J.; Rouw, E.; Kerssens, D.; den Bent van, H.: TOGAF the Open Group Architectural Framework: A Management Guide. Zaltbommel, Netherlands: Van Heren Publishing, 2007.

SARKAR & CHAKRABARTI 2007

Sarkar, P.; Chakrabarti, A.: Development of a Method for Assessing Design Creativity. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

SCHUH 1989

Schuh, G.: Gestaltung und Bewertung von Produktvarianten. Ein Beitrag zur systematischen Planung von Serienprodukten. Düsseldorf: VDI Verlag 1989.

SCHUH 2005

Schuh, G.: Produktkomplexität managen: Strategien - Methoden - Tools. München: Carl Hanser Verlag 2005. ISBN: 978-3446400436.

SHAH et al. 2009

Shah, A. A.; Schaefer, D.; Paredis, C. J. J.: Enabling Multi-View Modeling With SysML Profiles. In: Proceedings of the International Conference on Product Lifecycle Management PLM 09, Bath, UK, 06.-08.07.2009.

SHARMAN & YASSINE 2007

Sharman, D. M.; Yassine, A. A.: Architectural Valuation using the Design Structure Matrix and Real Options Theory. Concurrent Engineering 15 (2007) 2, pp. 157-173.

SHEA et al. 2010

Shea, K.; Ertelt, C.; Gmeiner, T.; Ameri, F.: Design-to-fabrication automation for the cognitive machine shop. Advanced Engineering Informatics 24 (2010) 3, pp. 251-268.

SIDDIQUE & ROSEN 1999

Siddique, Z.; Rosen, D. W.: Product Platform Design: a Graph Grammar Approach. In: Proceedings of the IDETC/CIE 1999 ASME 1999 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Las Vegas, 12.-16.9.1999. Las Vegas: ASME

SIMMONS 2008

Simmons, W. L.: A Framework for Decision Supporting in Systems Architecting. Dissertation, MIT, Cambridge 2008.

SIMON 1962

Simon, H. A.: The architecture of complexity. Proceedings of the American Philosophical Society 106 (1962) 6, pp. 467-482.

SIMON 1996

Simon, H. A.: The Sciences of the Artificial. Cambridge, MA: MIT Press, 1996.

SIMPSON et al. 2001

Simpson, T. W.; Maier, J. R. A.; Mistree, F.: Product Platform Design: Method and Application. In: Research in Engineering Design 13 (2001) 1, pp. 2-22.

SIMPSON 2004

Simpson, T. W.: Product platform design and customization: Status and promise. In: Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18 (2004) 1, pp. 3-20.

SIMPSON et al. 2006

Simpson, T. W.; Siddique, Z. Jiao, R.: Platform-Based Product Family Development. In: Simpson, T. W.; Jiao, J.; Tseng, M. (Eds.): Product Platform and Product Family Design: Methods and Applications. Berlin: Springer, 2006, pp. 1-15.

SOSA et al. 2004

Sosa, M. E.; Eppinger, S. D.; Rowles, C. M.: The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. Management Science 50 (2004) 12, pp. 1674-1689.

SOSA et al. 2005

Sosa, M. E.; Agrawal, A.; Eppinger, S. D.; Rowles, C. M.: A Network Approach to Define Modularity of Product Components. In: Proceedings of the IDETC/CIE 2005 ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Long Beach, 24.-28.09.2005. Long Beach: ASME, 2005.

SOSA et al. 2007

Sosa, M. E.; Eppinger, S. D.; Rowles, C. M.: A Network Approach to Define Modularity of Components in Complex Products. In: Journal of mechanical Design 129 (2007) 11.

STARK 2005

Stark, J.: Product Lifecycle Management: 21st Century Paradigm for Product Realisation. London: Springer 2005. ISBN: 1-8523-810-5.

STARLING & SHEA 2005

Starling, A. C.; Shea, K.: A Parallel Grammar for Simulation-Driven Mechanical Design Synthesis. Proceedings of the IDETC/CIE 2005 ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Long Beach, 24.-28.09.2005. Long Beach: ASME, 2005.

STEINMEIER 1999

Steinmeier, E.: Realisierung eines systemtechnischen Produktmodells - Einsatz in der PKW-Entwicklung. Dissertation, Technische Universität München, Lehrstuhl für Produktentwicklung, München (1999).

STEWARD 1981

Steward, D. V.: The Design Structure System: A Method for Managing the Design of Complex Systems. IEEE Transactions on Engineering Management 28 (1981) 3, pp. 71-74.

STONE & WOOD 2000

Stone, R. B.; Wood, K. L.: Development of a Functional Basis for Design. Journal of Mechanical Design 122 (2000) 4, pp. 359-370.

STRYKER & JACQUES 2009

Stryker, A. C.; Jacques, D. R.: Modularity versus Functionality: A Survey and Application. In: Proceedings of the 7th Annual Conference on Systems Engineering Research (CSER), 20.-22.04.2009, Loughborough, UK, 2009.

SUH 1990

Suh, N. P.: Axiomatic Design. New York: Oxford University Press, 1990.

SUH 1998

Suh, N. P.: Axiomatic Design Theory for Systems. In: Research in Engineering Design 10 (1998) 4, pp. 189-209.

SUH 2001

Suh, N. P.: Axiomatic Design - Advances and Applications. New York: Oxford University Press 2001. ISBN: 0-19-513466-4. (MIT-Pappalardo Series in Mechanical Engineering).

SUH et al. 2007

Suh, E. S.; De Weck, O. L.; Chang, D.: Flexible product platforms: framework and case study. In: Research in Engineering Design 18 (2007) 2, pp. 67-89.

TATE & NORDLUND 2001

Tate, D.; Nordlund, M.: Research Methods for Design Theory. In: Proceedings of the ASME 2001 Design Engineering Technical Conferences DETC 2001, Pittsburgh, 09.-12.09.2001, Pittsburgh, PA, 2001.

TERPENNY & MATHEW 2004

Terpenny, J.P.; Mathew, D.: Modeling Environment for Function-Based Conceptual Design. In: Proceedings of the ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (2004), 30th Design Automation Conference, 28.09.-02.10.2004, Salt Lake City, Utah, 2004.

THEVENOT & SIMPSON 2006

Thevenot, H. J.; Simpson, T. W.: Commonality indices for product family design: a detailed comparison. In: Journal of Engineering Design 17 (2006) 2, pp. 99-119.

THOMA 1975

Thoma, J.: Bond Graphs: Introduction and Applications. München: Elsevier, 1975, ISBN-13: 978-0080188812

THRAMBOULIDIS 2010

Thramboulidis, K.: The 3+1 SysML View-Model in Model Integrated Mechatronics. In: Journal of Software Engineering & Applications 3 (2010) 2, pp. 109-118.

TOMIYAMA et al. 2003

Tomiyama, T.; Takeda, H.; Yoshioka, M.; Shimomura, Y.: Abduction for creative design, In: Proceedings of the IDETC/CIE 2003 ASME 2003 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Chicago, 02.-06.09.2003. Chicago: ASME, 2003.

TOMIYAMA & SCHOTBORGH 2007

Tomiyama, T.; Schotborgh, W.: Yet another Model of Design Synthesis. In: Proceedings of the International Conference on Engineering Design, ICED’07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

TRIPATHY & EPPINGER 2007

Tripathy, A.; Eppinger, S. D.: A System Architecture Approach to Global Product Development. MIT Sloan School of Management Working Paper 4645-07 (2007) pp. 1-45.

TSUDA 1997

Tsuda, Y.: Concurrent Engineering Case Studies Applying QFD Models. Concurrent Engineering: Research and Applications 5 (1997) 4, pp. 337-345.

TURKI & SORIANO 2005

Turki, S.; Soriano, T.: A SysML Extension for Bond Graphs Support. In: Proceedings of the 5th International Conference on Technology and Automation ICTA, Thessaloniki, Greece, 15.-16.10.2005. Thessaloniki: Dept. of Automation, 2005.

ULRICH 1995

Ulrich, K. T.: The role of product architecture in the manufacturing firm. Research Policy 24 (1995) 3, pp. 419-440.

ULRICH & EPPINGER 2003

Ulrich, K. T.; Eppinger, S. D.: Product Design and Development. 3 Aufl. New York: Irwin/McGraw-Hill 2003.

UMEDA et al. 1990

Umeda, Y.; Takeda, H.; Tomiyama, T.; Yoshikawa, H.: Function, Behaviour, and Structure. In: Gero, J. S. (Ed.), Applications of Artificial Intelligence in Engineering V. Berlin: Springer, 1990.

UMEDA & TOMIYAMA 1995

Umeda, Y.; Tomiyama, T.: FBS Modeling: Modeling Scheme of Function for Conceptual Design. In: Proceedings of the 9th International Workshop on Qualitative Reasoning, Amsterdam, 16.-19.05.1995, pp. 271- 278.

UMEDA et al. 2005

Umeda, Y.; Kondoh, S.; Shimomura, Y.; Tomiyama, T.: Development of design methodology for upgradable products based on function-behavior-state modeling. In: Artificial Intelligence for Engineering Design, Analysis and Manufacturing 9 (2005) 3, pp. 161-182.

VDI 2206

Design methodology for mechatronic systems, VDI Fachbereich Produktentwicklung und Mechatronik (VDI 2206). 2004

VDI 2221

Systematic approach to the development and design of technical systems and products, VDI Fachbereich Produktentwicklung und Mechatronik (VDI 2221). 1993

VESTER 2001

Vester, F.: Die Kunst vernetzt zu denken. Ideen und Werkzeuge für einen neuen Umgang mit Komplexität. Stuttgart: DVA 2001. ISBN: 978-3421053084.

VON BERTALANFFY 1976

Von Bertalanffy, L.: General System Theory: Foundations, Development, Applications. New York: George Braziller 1976. ISBN: 978-0-8076-0453-3.

WANG et al. 2007

Wang, W.; Duffy, A.; Haffey, M.: A Post-positivism View of Function Behavior Structure. In: Proceedings of the International Conference on Engineering Design, ICED'07, Paris, France, 28.- 31.8.2007. Paris, France: Cite Des Sciences De L'Industrie 2007.

WARD 2007

Ward, A. C.: Lean Product and Process Development. Cambridge: Lean Enterprise Institute Inc. 2007. ISBN: 978-1-934109-13-7.

WATTS 2003

Watts, D. J.: Six Degrees: The Science of a Connected Age. New York: W.W. Norton & Co. 2003. ISBN: 978-0393325423.

WEBER 2005a

Weber, C.: CPM/PDD – An Extended Theoretical Approach to Modelling Products and Product Development Processes. In: Bley, H. et al. (Eds.): Proceedings of the 2nd German-Israeli Symposium on Advances in Methods and Systems for Development of Products and Processes 2005, Stuttgart, 07.-08.07.2005. Stuttgart: Fraunhofer-IRB-Verlag pp. 159-179.

WEBER 2005b

Weber, C.: What is "Complexity"? In: Proceedings of the International Conference of Engineering Design (ICED 05), Melbourne, Australia, 15. - 18.08.2005.

WEBER 2007

Weber, C.: Looking at "DFX" and "Product Maturity" from the Perspective of a New Approach to Modelling Product and Product Development Processes. In: Krause, F.-L. (Ed.): Proceedings of the 17th CIRP Design Conference in co-operation with Berliner Kreis, TU Berlin / Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK), 26.-28.03.2007. Berlin-Heidelberg: Springer pp. 85-104.

WEILKIENS 2008

Weilkiens, T.: Systems Engineering with SysML/UML: Modeling, Analysis, Design. Waltham, MA: Morgan Kaufmann, 2008.

WERTHNER 1994

Werthner, H.: Qualitative Reasoning: Modeling and the Generation of Behavior. Wien: Springer 1994. ISBN: 0-387-82579-7.

WEST 2001

West, D. B.: Introduction to Graph Theory. Upper Saddle River: Prentice Hall 2001. ISBN: 0-13-014400-2.

WHITNEY 2004

Whitney, D. E.: Physical limits to modularity. Engineering Systems Division Working paper. Cambrdige, MA: Massachusetts Institute of Technology, Engineering Systems Division, 2004.

WIESE & JOHN 2003

Wiese, P. R.; John, P.: Engineering Design in the Multi-Discipline Era: A Systems Approach. London: Professional Engineering Publishing Limited 2003. ISBN: 1-86058-347-4.

WILDEMANN 1999

Wildemann, H.: Produktklinik. Wertgestaltung von Produkten und Prozessen. Methoden und Fallbeispiele. München: TCW Transfer-Centrum-Verlag GmbH 1999.

WILDEMANN et al. 2007

Wildemann, H.; Ann, C.; Broy, M.; Günthner, W.A.; Lindemann, U.: Plagiatschutz. Handlungsspielräume der produzierenden Industrie gegen Produktpiraterie. München: TCW Transfer-Cenrum-Verlag GmbH 2007.

WILDEMANN 2008

Wildemann, H.: Komplexitätsmanagement in Vertrieb, Beschaffung, Produkt, Entwicklung und Produktion. München: TCW Transfer-Cenrum-Verlag GmbH 2008

WILLIAMS et al. 2007

Williams, D. B.; Allen, J. K.; Rosen, D. W.; Mistree, F.: Designing Platforms for Customizable Products and Processes in Markets of Non-Uniform Demand. In: Concurrent Engineering 15 (2007) 2, pp. 201-216.

WÖLKL & SHEA 2009

Wölkl, S.; Shea, K.: A computational product model for conceptual design using SysML. In: Proceedings of the ASME IDETC/CIE 2009, 30.08.-02.09.2009, San Diego, CA, 2009.

WOMACK & JONES 2006

Womack, J. P. ; Jones, D. T.: Lean Solutions: Wie Unternehmen und Kunden gemeinsam Probleme lösen. Frankfurt am Main: Campus Verlag 2006. ISBN: 978-3-593-38112-1.

WOMACK et al. 2007

Womack, J. P. ; Jones, D. T.; Roos, D.: The Machine That Changed the World: The Story of Lean Production-- Toyota's Secret Weapon in the Global Car Wars That Is Now Revolutionizing World Industry. New York: Free Press 2007. ISBN: 978-0743299794.

WOOD & GREER 2001

Wood, K. L., Greer, J. L.: Function-based synthesis methods in engineering design: state of the art, methods analysis, and visions for the future. In: Antonsson, Cagan: Formal engineering design synthesis. New York: Cambridge University Press, 2001.

WOUT et al. 2010

Wout van't, J.; Waage, M.; Hartman, H.; Stahlecker, M.; Hofmann, A.: The Integrated Architecture Framework Explained: Why, What, How. Berlin: Springer, 2010.

WYATT et al. 2008

Wyatt, D.; Wynn, D.; Clarkson, J.: Synthesis of Product Architectures Using a DSM/DMM-based Approach. In: Kreimeyer, M. et al. (Eds.): 10th International DSM Conference, Stockholm, 11.-12.11.2008. Munich: Hanser 2008, pp. 349-361. ISBN: 978-3-446-41825-7.

WYNN 2007

Wynn, D. C.: Model-based approaches to support process improvement in complex product development. Dissertation, University of Cambridge, Cambridge (2007).

YASSINE & WISSMANN 2007

Yassine, A. A.; Wissmann, L. A.: The Implications of Product Architecture on the Firm. Systems Engineering 10 (2007) 2, pp. 118-137.

ZACHMAN 1987

Zachman, J. A.: A Framework for Information Systems Architecture. In: IBM Systems Journal 26 (1987) 3, pp. 276-292.

ZANKER 1999

Zanker, W.: Situative Anpassung und Neukombination von Entwicklungsmethoden. Dissertation, Technische Universität München, Lehrstuhl für Konstruktion (1999).

ZHA & LIU 2005

Zha, X. F.; Liu, C. L.: Information Modeling for Computer-Supported MEMS Product Design and Development. In: Proceedings of The 15th International CIRP Design Seminar 2005, 22.-26.05.2005, Shnghai, China, 2005.

ZHANG & GERSHENSON 2001

Zhang, Y.; Gershenson, J. K.: An Initial Study of Direct Relationships between Life-Cycle Modularity and Life-Cycle Cost. In: Concurrent Engineering 11 (2003) 2, pp. 121-128.

ZIMMERMANN & STACHE 2001

Zimmermann, W.; Stache, U.: Operations Research: Quantitative Methoden zur Entscheidungsvorbereitung. München: R. Oldenbourg 2001. ISBN: 3-486-25816-8.

ZIRKLER 2010

Zirkler, S.: Transdisziplinäres Zielkostenmanagement komplexer mechatronischer Produkte. Dissertation, Technische Universität München, Lehrstuhl für Produktentwicklung, München (2010).

ZÜST 1997

Züst, R.: Einstieg ins Systems Engineering: Systematisch denken, handeln und umsetzen. Zürich: Verlag Industrielle Organisation 1997. ISBN: 3-85743-990-4.

ZWICKY 1969

Zwicky, F.: Discovery, Invention, Research: Through the Morphological Approach. Toronto: The Macmillan Company, 1969.

11. List of dissertations

Lehrstuhl für Produktentwicklung
Technische Universität München, Boltzmannstraße 15, 85748 Garching
Dissertationen betreut von

- Prof. Dr.-Ing. W. Rodenacker,
- Prof. Dr.-Ing. K. Ehrlenspiel und
- Prof. Dr.-Ing. U. Lindemann

D1 COLLIN, H.:
Entwicklung eines Einwalzenkalanders nach einer systematischen Konstruktionsmethode. München: TU, Diss. 1969.

D2 OTT, J.:
Untersuchungen und Vorrichtungen zum Offen-End-Spinnen.
München: TU, Diss. 1971.

D3 STEINWACHS, H.:
Informationsgewinnung an bandförmigen Produkten für die Konstruktion der Produktmaschine.
München: TU, Diss. 1971.

D4 SCHMETTOW, D.:
Entwicklung eines Rehabilitationsgerätes für Schwerstkörperbehinderte.
München: TU, Diss. 1972.

D5 LUBITZSCH, W.:
Die Entwicklung eines Maschinensystems zur Verarbeitung von chemischen Endlosfasern.
München: TU, Diss. 1974.

D6 SCHEITENBERGER, H.:
Entwurf und Optimierung eines Getriebesystems für einen Rotationsquerschneider mit allgemeingültigen Methoden.
München: TU, Diss. 1974.

D7 BAUMGARTH, R.:
Die Vereinfachung von Geräten zur Konstanthaltung physikalischer Größen.
München: TU, Diss. 1976.

D8 MAUDERER, E.:
Beitrag zum konstruktionsmethodischen Vorgehen durchgeführt am Beispiel eines Hochleistungsschalter-Antriebs.
München: TU, Diss. 1976.

D9 SCHÄFER, J.:
Die Anwendung des methodischen Konstruierens auf verfahrenstechnische Aufgabenstellungen.
München: TU, Diss. 1977.

D10 WEBER, J.:
Extruder mit Feststoffpumpe – Ein Beitrag zum Methodischen Konstruieren.
München: TU, Diss. 1978.

D11 HEISIG, R.:
Längencodierer mit Hilfsbewegung.
München: TU, Diss. 1979.

D12 KIEWERT, A.:
Systematische Erarbeitung von Hilfsmitteln zum kostenarmen Konstruieren.
München: TU, Diss. 1979.

D13 LINDEMANN, U.:
Systemtechnische Betrachtung des Konstruktionsprozesses unter besonderer Berücksichtigung der Herstellkostenbeeinflussung beim Festlegen der Gestalt.
Düsseldorf: VDI-Verlag 1980. (Fortschritt-Berichte der VDI-Zeitschriften Reihe 1, Nr. 60).
Zugl. München: TU, Diss. 1980.

D14 NJOYA, G.:
Untersuchungen zur Kinematik im Wälzlager bei synchron umlaufenden Innen- und Außenringen.
Hannover: Universität, Diss. 1980.

D15 HENKEL, G.:
Theoretische und experimentelle Untersuchungen ebener konzentrisch gewellter Kreisringmembranen.
Hannover: Universität, Diss. 1980.

D16 BALKEN, J.:
Systematische Entwicklung von Gleichlaufgelenken.
München: TU, Diss. 1981.

D17 PETRA, H.:
Systematik, Erweiterung und Einschränkung von Lastausgleichslösungen für Standgetriebe mit zwei Leistungswegen – Ein Beitrag zum methodischen Konstruieren.
München: TU, Diss. 1981.

D18 BAUMANN, G.:
Ein Kosteninformationssystem für die Gestaltungsphase im Betriebsmittelbau.
München: TU, Diss. 1982.

D19 FISCHER, D.:
Kostenanalyse von Stirnzahnrädern. Erarbeitung und Vergleich von Hilfsmitteln zur Kostenfrüherkennung.
München: TU, Diss. 1983.

D20 AUGUSTIN, W.:
Sicherheitstechnik und Konstruktionsmethodiken – Sicherheitsgerechtes Konstruieren.
Dortmund: Bundesanstalt für Arbeitsschutz 1985. Zugl. München: TU, Diss. 1984.

D21 RUTZ, A.:
Konstruieren als gedanklicher Prozess.
München: TU, Diss. 1985.

D22 SAUERMANN, H. J.:
Eine Produktkostenplanung für Unternehmen des Maschinenbaues.
München: TU, Diss. 1986.

D23 HAFNER, J.:
Entscheidungshilfen für das kostengünstige Konstruieren von Schweiß- und Gussgehäusen.
München: TU, Diss. 1987.

D24 JOHN, T.:
Systematische Entwicklung von homokinetischen Wellenkupplungen.
München: TU, Diss. 1987.

D25 FIGEL, K.:
Optimieren beim Konstruieren.
München: Hanser 1988. Zugl. München: TU, Diss. 1988 u. d. T.: Figel, K.: Integration automatisierter Optimierungsverfahren in den rechnerunterstützten Konstruktionsprozess.

Reihe Konstruktionstechnik München

D26 TROPSCHUH, P. F.:
Rechnerunterstützung für das Projektieren mit Hilfe eines wissensbasierten Systems.
München: Hanser 1989. (Konstruktionstechnik München, Band 1). Zugl. München: TU, Diss. 1988 u. d. T.: Tropschuh, P. F.: Rechnerunterstützung für das Projektieren am Beispiel Schiffsgetriebe.

D27 PICKEL, H.:
Kostenmodelle als Hilfsmittel zum Kostengünstigen Konstruieren.
München: Hanser 1989. (Konstruktionstechnik München, Band 2). Zugl. München: TU, Diss. 1988.

D28 KITTSTEINER, H.-J.:
Die Auswahl und Gestaltung von kostengünstigen Welle-Nabe-Verbindungen.
München: Hanser 1990. (Konstruktionstechnik München, Band 3). Zugl. München: TU, Diss. 1989.

D29 HILLEBRAND, A.:
Ein Kosteninformationssystem für die Neukonstruktion mit der Möglichkeit zum Anschluss an ein CAD-System.
München: Hanser 1991. (Konstruktionstechnik München, Band 4). Zugl. München: TU, Diss. 1990.

D30 DYLLA, N.:
Denk- und Handlungsabläufe beim Konstruieren.
München: Hanser 1991. (Konstruktionstechnik München, Band 5). Zugl. München: TU, Diss. 1990.

D31 MÜLLER, R.
Datenbankgestützte Teileverwaltung und Wiederholteilsuche.
München: Hanser 1991. (Konstruktionstechnik München, Band 6). Zugl. München: TU, Diss. 1990.

D32 NEESE, J.:
Methodik einer wissensbasierten Schadenanalyse am Beispiel Wälzlagerungen.
München: Hanser 1991. (Konstruktionstechnik München, Band 7). Zugl. München: TU, Diss. 1991.

D33 SCHAAL, S.:
Integrierte Wissensverarbeitung mit CAD – Am Beispiel der konstruktionsbegleitenden Kalkulation.
München: Hanser 1992. (Konstruktionstechnik München, Band 8). Zugl. München: TU, Diss. 1991.

D34 BRAUNSPERGER, M.:
Qualitätssicherung im Entwicklungsablauf – Konzept einer präventiven Qualitätssicherung für die Automobilindustrie.
München: Hanser 1993. (Konstruktionstechnik München, Band 9). Zugl. München: TU, Diss. 1992.

D35 FEICHTER, E.:
Systematischer Entwicklungsprozess am Beispiel von elastischen Radialversatzkupplungen.
München: Hanser 1994. (Konstruktionstechnik München, Band 10). Zugl. München: TU, Diss. 1992.

D36 WEINBRENNER, V.:
Produktlogik als Hilfsmittel zum Automatisieren von Varianten- und Anpassungskonstruktionen.
München: Hanser 1994. (Konstruktionstechnik München, Band 11). Zugl. München: TU, Diss. 1993.

D37 WACH, J. J.:
Problemspezifische Hilfsmittel für die Integrierte Produktentwicklung.
München: Hanser 1994. (Konstruktionstechnik München, Band 12). Zugl. München: TU, Diss. 1993.

D38 LENK, E.:
Zur Problematik der technischen Bewertung.
München: Hanser 1994. (Konstruktionstechnik München, Band 13). Zugl. München: TU, Diss. 1993.

D39 STUFFER, R.:
Planung und Steuerung der Integrierten Produktentwicklung.
München: Hanser 1994. (Konstruktionstechnik München, Band 14). Zugl. München: TU, Diss. 1993.

D40 SCHIEBELER, R.:
Kostengünstig Konstruieren mit einer rechnergestützten Konstruktionsberatung.
München: Hanser 1994. (Konstruktionstechnik München, Band 15). Zugl. München: TU, Diss. 1993.

D41 BRUCKNER, J.:
Kostengünstige Wärmebehandlung durch Entscheidungsunterstützung in Konstruktion und Härterei.
München: Hanser 1994. (Konstruktionstechnik München, Band 16). Zugl. München: TU, Diss. 1993.

D42 WELLNIAK, R.:
Das Produktmodell im rechnerintegrierten Konstruktionsarbeitsplatz.
München: Hanser 1994. (Konstruktionstechnik München, Band 17). Zugl. München: TU, Diss. 1994.

D43 SCHLÜTER, A.:
Gestaltung von Schnappverbindungen für montagegerechte Produkte.
München: Hanser 1994. (Konstruktionstechnik München, Band 18). Zugl. München: TU, Diss. 1994.

D44 WOLFRAM, M.:
Feature-basiertes Konstruieren und Kalkulieren.
München: Hanser 1994. (Konstruktionstechnik München, Band 19). Zugl. München: TU, Diss. 1994.

D45 STOLZ, P.:
Aufbau technischer Informationssysteme in Konstruktion und Entwicklung am Beispiel eines elektronischen Zeichnungsarchives.
München: Hanser 1994. (Konstruktionstechnik München, Band 20). Zugl. München: TU, Diss. 1994.

D46 STOLL, G.:
Montagegerechte Produkte mit feature-basiertem CAD.
München: Hanser 1994. (Konstruktionstechnik München, Band 21). Zugl. München: TU, Diss. 1994.

D47 STEINER, J. M.:
Rechnergestütztes Kostensenken im praktischen Einsatz.
Aachen: Shaker 1996. (Konstruktionstechnik München, Band 22). Zugl. München: TU, Diss. 1995.

D48 HUBER, T.:
Senken von Montagezeiten und -kosten im Getriebebau.
München: Hanser 1995. (Konstruktionstechnik München, Band 23). Zugl. München: TU, Diss. 1995.

D49 DANNER, S.:
Ganzheitliches Anforderungsmanagement für marktorientierte Entwicklungsprozesse.
Aachen: Shaker 1996. (Konstruktionstechnik München, Band 24). Zugl. München: TU, Diss. 1996.

D50 MERAT, P.:
Rechnergestützte Auftragsabwicklung an einem Praxisbeispiel.
Aachen: Shaker 1996. (Konstruktionstechnik München, Band 25). Zugl. München: TU, Diss. 1996 u. d. T.: MERAT, P.: Rechnergestütztes Produktleitsystem

D51 AMBROSY, S.:
Methoden und Werkzeuge für die integrierte Produktentwicklung.
Aachen: Shaker 1997. (Konstruktionstechnik München, Band 26). Zugl. München: TU, Diss. 1996.

D52 GIAPOULIS, A.:
Modelle für effektive Konstruktionsprozesse.
Aachen: Shaker 1998. (Konstruktionstechnik München, Band 27). Zugl. München: TU, Diss. 1996.

D53 STEINMEIER, E.:
Realisierung eines systemtechnischen Produktmodells – Einsatz in der Pkw-Entwicklung
Aachen: Shaker 1998. (Konstruktionstechnik München, Band 28). Zugl. München: TU, Diss. 1998.

D54 KLEEDÖRFER, R.:
Prozess- und Änderungsmanagement der Integrierten Produktentwicklung.
Aachen: Shaker 1998. (Konstruktionstechnik München, Band 29). Zugl. München: TU, Diss. 1998.

D55 GÜNTHER, J.:
Individuelle Einflüsse auf den Konstruktionsprozess.
Aachen: Shaker 1998. (Konstruktionstechnik München, Band 30). Zugl. München: TU, Diss. 1998.

D56 BIERSACK, H.:
Methode für Krafteinleitungsstellenkonstruktion in Blechstrukturen.
München: TU, Diss. 1998.

D57 IRLINGER, R.:
Methoden und Werkzeuge zur nachvollziehbaren Dokumentation in der Produktentwicklung.
Aachen: Shaker 1998. (Konstruktionstechnik München, Band 31). Zugl. München: TU, Diss. 1999.

D58 EILETZ, R.:
Zielkonfliktmanagement bei der Entwicklung komplexer Produkte – am Bsp. PKW-Entwicklung.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 32). Zugl. München: TU, Diss. 1999.

D59 STÖSSER, R.:
Zielkostenmanagement in integrierten Produkterstellungsprozessen.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 33). Zugl. München: TU, Diss. 1999.

D60 PHLEPS, U.:
Recyclinggerechte Produktdefinition – Methodische Unterstützung für Upgrading und Verwertung.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 34). Zugl. München: TU, Diss. 1999.

D61 BERNARD, R.:
Early Evaluation of Product Properties within the Integrated Product Development.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 35). Zugl. München: TU, Diss. 1999.

D62 ZANKER, W.:
Situative Anpassung und Neukombination von Entwicklungsmethoden.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 36). Zugl. München: TU, Diss. 1999.

Reihe Produktentwicklung München

D63 ALLMANSBERGER, G.:
Erweiterung der Konstruktionsmethodik zur Unterstützung von Änderungsprozessen in der Produktentwicklung.
München: Dr. Hut 2001. (Produktentwicklung München, Band 37). Zugl. München: TU, Diss. 2000.

D64 ASSMANN, G.:
Gestaltung von Änderungsprozessen in der Produktentwicklung.
München: Utz 2000. (Produktentwicklung München, Band 38). Zugl. München: TU, Diss. 2000.

D65 BICHLMAIER, C.:
Methoden zur flexiblen Gestaltung von integrierten Entwicklungsprozessen.
München: Utz 2000. (Produktentwicklung München, Band 39). Zugl. München: TU, Diss. 2000.

D66 DEMERS, M. T.
Methoden zur dynamischen Planung und Steuerung von Produktentwicklungsprozessen.
München: Dr. Hut 2000. (Produktentwicklung München, Band 40). Zugl. München: TU, Diss. 2000.

D67 STETTER, R.:
Method Implementation in Integrated Product Development.
München: Dr. Hut 2000. (Produktentwicklung München, Band 41). Zugl. München: TU, Diss. 2000.

D68 VIERTLBÖCK, M.:
Modell der Methoden- und Hilfsmitteleinführung im Bereich der Produktentwicklung.
München: Dr. Hut 2000. (Produktentwicklung München, Band 42). Zugl. München: TU, Diss. 2000.

D69 COLLIN, H.:
Management von Produkt-Informationen in kleinen und mittelständischen Unternehmen.
München: Dr. Hut 2001. (Produktentwicklung München, Band 43). Zugl. München: TU, Diss. 2001.

D70 REISCHL, C.:
Simulation von Produktkosten in der Entwicklungsphase.
München: Dr. Hut 2001. (Produktentwicklung München, Band 44). Zugl. München: TU, Diss. 2001.

D71 GAUL, H.-D.:
Verteilte Produktentwicklung - Perspektiven und Modell zur Optimierung.
München: Dr. Hut 2001. (Produktentwicklung München, Band 45). Zugl. München: TU, Diss. 2001.

D72 GIERHARDT, H.:
Global verteilte Produktentwicklungsprojekte – Ein Vorgehensmodell auf der operativen Ebene.
München: Dr. Hut 2002. (Produktentwicklung München, Band 46). Zugl. München: TU, Diss. 2001.

D73 SCHOEN, S.:
Gestaltung und Unterstützung von Community of Practice.
München: Utz 2000. (Produktentwicklung München, Band 47). Zugl. München: TU, Diss. 2000.

D74 BENDER, B.:
Zielorientiertes Kooperationsmanagement.
München: Dr. Hut 2001. (Produktentwicklung München, Band 48). Zugl. München: TU, Diss. 2001.

D75 SCHWANKL, L.:
Analyse und Dokumentation in den frühen Phasen der Produktentwicklung.
München: Dr. Hut 2002. (Produktentwicklung München, Band 49). Zugl. München: TU, Diss. 2002.

D76 WULF, J.:
Elementarmethoden zur Lösungssuche.
München: Dr. Hut 2002. (Produktentwicklung München, Band 50). Zugl. München: TU, Diss. 2002.

D77 MÖRTL, M.:
Entwicklungsmanagement für langlebige, upgradinggerechte Produkte.
München: Dr. Hut 2002. (Produktentwicklung München, Band 51). Zugl. München: TU, Diss. 2002.

D78 GERST, M.:
Strategische Produktentscheidungen in der integrierten Produktentwicklung.
München: Dr. Hut 2002. (Produktentwicklung München, Band 52). Zugl. München: TU, Diss. 2002.

D79 AMFT, M.:
Phasenübergreifende bidirektionale Integration von Gestaltung und Berechnung.
München: Dr. Hut 2003. (Produktentwicklung München, Band 53). Zugl. München: TU, Diss. 2002.

D80 FÖRSTER, M.:
Variantenmanagement nach Fusionen in Unternehmen des Anlagen- und Maschinenbaus.
München: TU, Diss. 2003.

D81 GRAMANN, J.:
Problemmodelle und Bionik als Methode.
München: Dr. Hut 2004. (Produktentwicklung München, Band 55). Zugl. München: TU, Diss. 2004.

D82 PULM, U.:
Eine systemtheoretische Betrachtung der Produktentwicklung.
München: Dr. Hut 2004. (Produktentwicklung München, Band 56). Zugl. München: TU, Diss. 2004.

D83 HUTTERER, P.:
Reflexive Dialoge und Denkbausteine für die methodische Produktentwicklung.
München: Dr. Hut 2005. (Produktentwicklung München, Band 57). Zugl. München: TU, Diss. 2005.

D84 FUCHS, D.:
Konstruktionsprinzipien für die Problemanalyse in der Produktentwicklung.
München: Dr. Hut 2006. (Produktentwicklung München, Band 58). Zugl. München: TU, Diss. 2005.

D85 PACHE, M.:
Sketching for Conceptual Design.
München: Dr. Hut 2005. (Produktentwicklung München, Band 59). Zugl. München: TU, Diss. 2005.

D86 BRAUN, T.:
Methodische Unterstützung der strategischen Produktplanung in einem mittelständisch geprägten Umfeld.
München: Dr. Hut 2005. (Produktentwicklung München, Band 60). Zugl. München: TU, Diss. 2005.

D87 JUNG, C.:
Anforderungsklärung in interdisziplinärer Entwicklungsumgebung.
München: Dr. Hut 2006. (Produktentwicklung München, Band 61). Zugl. München: TU, Diss. 2006.

D88 HEßLING, T.:
Einführung der Integrierten Produktpolitik in kleinen und mittelständischen Unternehmen.
München: Dr. Hut 2006. (Produktentwicklung München, Band 62). Zugl. München: TU, Diss. 2006.

D89 STRICKER, H.:
Bionik in der Produktentwicklung unter der Berücksichtigung menschlichen Verhaltens.
München: Dr. Hut 2006. (Produktentwicklung München, Band 63). Zugl. München: TU, Diss. 2006.

D90 NIßL, A.:
Modell zur Integration der Zielkostenverfolgung in den Produktentwicklungsprozess.
München: Dr. Hut 2006. (Produktentwicklung München, Band 64). Zugl. München: TU, Diss. 2006.

D91 MÜLLER, F.:
Intuitive digitale Geometriemodellierung in frühen Entwicklungsphasen.
München: Dr. Hut 2007. (Produktentwicklung München, Band 65). Zugl. München: TU, Diss. 2006.

D92 ERDELL, E.:
Methodenanwendung in der Hochbauplanung – Ergebnisse einer Schwachstellenanalyse.
München: Dr. Hut 2006. (Produktentwicklung München, Band 66). Zugl. München: TU, Diss. 2006.

D93 GAHR, A.:
Pfadkostenrechnung individualisierter Produkte.
München: Dr. Hut 2006. (Produktentwicklung München, Band 67). Zugl. München: TU, Diss. 2006.

D94 RENNER, I.:
Methodische Unterstützung funktionsorientierter Baukastenentwicklung am Beispiel Automobil.
München: Dr. Hut 2007 (Reihe Produktentwicklung) Zugl. München: TU, Diss. 2007.

D95 PONN, J.:
Situative Unterstützung der methodischen Konzeptentwicklung technischer Produkte.
München: Dr. Hut 2007 (Reihe Produktentwicklung) Zugl. München: TU, Diss. 2007.

D96 HERFELD, U.:
Matrix-basierte Verknüpfung von Komponenten und Funktionen zur Integration von Konstruktion und numerischer Simulation.
München: Dr. Hut 2007. (Produktentwicklung München, Band 70). Zugl. München: TU, Diss. 2007.

D97 SCHNEIDER, S.:
Model for the evaluation of engineering design methods.
München: Dr. Hut 2008 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2007.

D98 FELGEN, L.:
Systemorientierte Qualitätssicherung für mechatronische Produkte.
München: Dr. Hut 2007 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2007.

D99 GRIEB, J.:
Auswahl von Werkzeugen und Methoden für verteilte Produktentwicklungsprozesse.
München: Dr. Hut 2007 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2007.

D100 MAURER, M.:
Structural Awareness in Complex Product Design.
München: Dr. Hut 2007 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2007.

D101 BAUMBERGER, C.:
Methoden zur kundenspezifischen Produktdefinition bei individualisierten Produkten.
München: Dr. Hut 2007 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2007.

D102 KEIJZER, W.:
Wandlungsfähigkeit von Entwicklungsnetzwerken – ein Modell am Beispiel der Automobilindustrie.
München: Dr. Hut 2007 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2007.

D103 LORENZ, M.:
Handling of Strategic Uncertainties in Integrated Product Development.
München: Dr. Hut 2009 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2008.

D104 KREIMEYER, M.:
Structural Measurement System for Engineering Design Processes.
München: Dr. Hut 2010 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2009.

D105 DIEHL, H.:
Systemorientierte Visualisierung disziplinübergreifender Entwicklungsabhängigkeiten mechatronischer Automobilsysteme.
München: Dr. Hut 2009 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2009.

D106 DICK, B.:
Untersuchung und Modell zur Beschreibung des Einsatzes bildlicher Produktmodelle durch Entwicklerteams in der Lösungssuche.
München: Dr. Hut 2009 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2009.

D107 GAAG, A.:
Entwicklung einer Ontologie zur funktionsorientierten Lösungssuche in der Produktentwicklung.
München: Dr. Hut 2010 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2010.

D108 ZIRKLER, S.:
Transdisziplinäres Zielkostenmanagement komplexer mechatronischer Produkte.
München: Dr. Hut 2010 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2010.

D109 LAUER, W.:
Integrative Dokumenten- und Prozessbeschreibung in dynamischen Produktentwicklungsprozessen.
München: Dr. Hut 2010 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2010.

D110 MEIWALD, T.:
Konzepte zum Schutz vor Produktpiraterie und unerwünschtem Know-how-Abfluss.
München: Dr. Hut 2011 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2011.

D111 ROELOFSEN, J.:
Situationsspezifische Planung von Produktentwicklungsprozessen.
München: Dr. Hut 2011 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2011.

D112 PETERMANN, M.:
Schutz von Technologiewissen in der Investitionsgüterindustrie.
München: Dr. Hut 2011 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2011.

D113 GORBEA, C.:
Vehicle Architecture and Lifecycle Cost Analysis in a New Age of Architectural Competition.
München: Dr. Hut 2011 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2011.

D114 FILOUS, M.:
Lizenzierungsgerechte Produktentwicklung – Ein Leitfaden zur Integration lizenzierungsrelevanter Aktivitäten in Produktentstehungsprozessen des Maschinen- und Anlagenbaus.
München: Dr. Hut 2011 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2011.

D115 ANTON, T.:
Entwicklungs- und Einführungsmethodik für das Projektierungswerkzeug Pneumatiksimulation.
München: Dr. Hut 2011 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2011.

D116 KESPER, H.:
Gestaltung von Produktvariantenspektren mittels matrixbasierter Methoden.
München: Dr. Hut 2012 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2012.

D117 KIRSCHNER, R.:
Methodische Offene Produktentwicklung.
München: TU, Diss. 2012.

D118 HEPPERLE, C.:
Planung lebenszyklusgerechter Leistungsbündel.
München: Dr. Hut 2013 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2013.

D119 HELLENBRAND, D.:
Transdisziplinäre Planung und Synchronisation mechatronischer Produktentwicklungsprozesse.
München: Dr. Hut 2013 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2013.

D120 EBERL, T.:
Charakterisierung und Gestaltung des Fahr-Erlebens der Längsführung von Elektrofahrzeugen.
München: TU, Diss. 2014.

D121 KAIN, A.:
Methodik zur Umsetzung der Offenen Produktentwicklung.
München: Dr. Hut 2014 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D122 ILIE, D.:
Systematisiertes Ziele- und Anforderungsmanagement in der Fahrzeugentwicklung.
München: Dr. Hut 2013 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2013.

D123 HELTEN, K.:
Einführung von Lean Development in mittelständische Unternehmen - Beschreibung, Erklärungsansatz und Handlungsempfehlungen.
München: Dr. Hut 2015 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D124 SCHRÖER, B.:
Lösungskomponente Mensch. Nutzerseitige Handlungsmöglichkeiten als Bausteine für die kreative Entwicklung von Interaktionslösungen.
München: TU, Diss. 2014.

D125 KORTLER, S.:
Absicherung von Eigenschaften komplexer und variantenreicher Produkte in der Produktentwicklung.
München: Dr. Hut 2014 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D126 KOHN, A.:
Entwicklung einer Wissensbasis für die Arbeit mit Produktmodellen.
München: Dr. Hut 2014 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D127 FRANKE, S.:
Strategieorientierte Vorentwicklung komplexer Produkte – Prozesse und Methoden zur zielgerichteten Komponentenentwicklung am Beispiel Pkw.
Göttingen: Cuvillier, E 2014. Zugl. München: TU, Diss. 2014.

D128 HOOSHMAND, A.:
Solving Engineering Design Problems through a Combination of Generative Grammars and Simulations.
München: Dr. Hut 2014 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D129 KISSEL, M.:
Mustererkennung in komplexen Produktportfolios.
München: TU, Diss. 2014.

D130 NIES, B.:
Nutzungsgerechte Dimensionierung des elektrischen Antriebssystems für Plug-In Hybride.
München: TU, Diss. 2014.

D131 KIRNER, K.:
Zusammenhang zwischen Leistung in der Produktentwicklung und Variantenmanagement – Einflussmodell und Analysemethode.
München: Dr. Hut 2014 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D132 BIEDERMANN, W.:
A minimal set of network metrics for analysing mechatronic product concepts.
München: TU, Diss. 2015.

D133 SCHENKL, S.:
Wissensorientierte Entwicklung von Produkt-Service-Systemen.
München: TU, Diss. 2015.

D134 SCHRIEVERHOFF, P.:
Valuation of Adaptability in System Architecture.
München: Dr. Hut 2015 (Reihe Produktentwicklung). Zugl. München: TU, Diss. 2014.

D135 METZLER, T.:
Models and Methods for the Systematic Integration of Cognitive Functions into Product Concepts.
München: Dr. Hut 2016 (Reihe Produktentwicklung).

D136 DEUBZER, F.:
A Method for Product Architecture Management in Early Phases of Product Development.
München: TU, Diss. 2016.

D137 SCHÖTTL, F.:
Komplexität in sozio-technischen Systemen - Methodik für die komplexitätsgerechte Systemgestaltung in der Automobilproduktion.
TU München: 2015. (als Dissertation eingereicht)

D138 BRANDT, L. S.:
Architekturgesteuerte Elektrik/Elektronik Baukastenentwicklung im Automobil
TU München: 2015. (als Dissertation eingereicht)

D139 BAUER, W.:
Planung und Entwicklung änderungsrobuster Plattformarchitekturen
TU München: 2015. (als Dissertation eingereicht)

D140 ELEZI, F.:
Supporting the Design of Management Control Systems In Engineering Companies from Management Cybernetics Perspective
München: TU, Diss. 2015.

D141 BEHNCKE, F. G. H.:
Beschaffungsgerechte Produktentwicklung – Abstimmung von Produktarchitektur und Liefernetzwerk in frühen Phasen der Entwicklung
TU München: 2015. (als Dissertation eingereicht)

D142 ÖLMEZ, M.:
Individuelle Unterstützung von Entscheidungsprozessen bei der Entwicklung innovativer Produkte.
TU München: 2016. (als Dissertation eingereicht)

D143 SAUCKEN, C. C. V.:
Entwicklerzentrierte Hilfsmittel zum Gestalten von Nutzererlebnissen.
TU München: 2016. (als Dissertation eingereicht)

D144 KASPEREK, D.:
Structure-based System Dynamics Analysis of Engineering Design Processes
TU München: 2016. (als Dissertation eingereicht)

D145 LANGER, S. F.:
Kritische Änderungen in der Produktentwicklung – Analyse und Maßnahmenableitung
TU München: 2016. (als Dissertation eingereicht)

D146 HERBERG, A. P.:
Planung und Entwicklung multifunktionaler Kernmodule in komplexen Systemarchitekturen und –portfolios – Methodik zur Einnahme einer konsequent modulzentrierten Perspektive
TU München: 2016. (als Dissertation eingereicht)

D147 HASHEMI FARZANEH, H.:
Bio-inspired design: Ideation in collaboration between mechanical engineers and biologists
TU München: 2016. (als Dissertation eingereicht)

D148 HELMS, M. K.:
Biologische Publikationen als Ideengeber für das Lösen technischer Probleme in der Bionik
TU München: 2016. (als Dissertation eingereicht)