Lehrstuhl für Produktentwicklung
der Technischen Universität München

Structural Awareness in Complex Product Design

Maik S. Maurer

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. Andrew Kusiak, Ph. D. University of Iowa/USA

Die Dissertation wurde am 27.06.2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 11.10.2007 angenommen.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
http://dnb.d-nb.de abrufbar.

ISBN 978-3-89963-632-1

Sternstr. 18, 80538 München
Tel.: 089/66060798
www.dr.hut-verlag.de

1. Auflage 2007

Druck und Bindung: fm-kopierbar, München (www.fm-kopierbar.de)

FOREWORD OF THE EDITOR

Problem

Complexity in product design steadily increases and represents a major challenge for any enterprise's sustainable market success. Hereby, complexity arises from different areas, e.g. markets, products, processes or organizations. Whereas already the amount of system elements and linkages represent a high complexity, especially the occurrence of system changes is difficult to handle. All mentioned areas are highly interconnected and e.g. the market request for product customization or an increased number of product variants can result in the need for extensive changes to the product program or the process of product creation. As changes originating from external areas as the market can hardly be avoided by enterprises, they have to handle change effects in their complex and highly interlinked products and processes. Considered from a system perspective, the structures emerging from system elements and their linkages largely contribute to the system characteristics and behavior. Thus, structural constellations in complex systems have to be considered and actively designed. This can improve the management and the specific design of such systems. So far, structural constellations are rarely used for managing complex systems and suitable methods and tools are not available in product design. However, the assessment and active creation of structural constellations can be a key to improved complexity management and can therefore provide competitive advantages for enterprises in the future.

Objectives

The described situation requires a methodical approach to analyze, to control, and to improve complex constellations in product development by focusing on the underlying structural dependencies. Furthermore, the approach allows for the consideration of multiple domains, as most complex situations and systems do not comprise single domains only. Such multiple domain consideration could enable the improvement of product development in spite of increasing complexity and can also be the basis for the implementation of upcoming business strategies that focus on enhanced product customization.

Results

New possibilities exceed the actual methods applied for complexity management in product design: the access of complex dependency networks by the Multiple-Domain Matrix closes the gap between available matrix-based methods and provides the so far missing generic model for the analysis, control and optimization of complex systems even if they comprise multiple aspects. The approach integrates established matrix-based methods and supports the definition of pertinent system aspects. The acquisition of information about complex systems is methodically supported and can be used for the extraction of experience knowledge. In addition, the possibilities of visualization complex networks have been systematically enhanced and assigned to specific tasks in complexity management. This helps to increase

system understanding for designers and allows for user interaction. In order to interpret structural constellations in complex systems the analysis criteria actually available have been enhanced. Based on these analysis criteria methodical applications of structure consideration on design are provided. On the one hand, methods for the improved management of complex structures are provided. These methods can enable designers to better cope with required system changes. The methods meant for the improvement of system design help users to implement a structural design that contributes to the desired system behavior. The entire approach presented in the thesis facilitates users in the management of complexity in product design by focusing on structural constellations of system dependencies and the simultaneous consideration of all relevant aspects.

Conclusions for industrial applications

The increasing complexity in almost all aspects of product design turned out to be a considerable handicap for enterprises. However, an improved complexity management can provide competitive advantages as e.g. market offers can be improved and product changes become more flexible and faster. It is important to mention that the knowledge about controlling complexity can not be easily copied by competitors – this knowledge is not an integral part of sold products and does not leave the company. For this reason, successful complexity management can be a core competence for enterprises and contribute to sustained market success.

Conclusions for scientific researches

The Multiple-Domain Matrix represents the methodical superstructure of the matrix-based approaches on structure-based complexity management. Due to this standardization it becomes possible to transfer available methods of analysis or optimization from one specific matrix approach to several more application scenarios. The consequent integration of multiple domains extended possible considerations of structural constellations to more comprehensive use cases. The method of Multiple-Domain Mapping provides a mathematical description for deriving specific system views that have so far only been created intuitively. The explicit description can serve for future enhancement of system structures by quantification with attributes, e.g. cost information. The criteria that are available for the analysis of system structures have been traced back to their origins in graph theory. This allows for the subsequent supplementation of analysis criteria and the improvement of structure interpretation. In future steps the systematic compilation of criteria has to be examined for completeness and for possibly combined criteria application. An improved assignment of interpretation occurrence of analysis criteria in user-defined structures can further increase the potential of structure-based complexity management in product design.

Garching, November 2007

Prof. Dr.-Ing. Udo Lindemann

Institute of Product Development
Technische Universität München

ACKNOWLEDGEMENT

This work results from my occupation as a scientific researcher at the Institute of Product Development at the Technische Universität München from December 2002 until November 2007.

Special thanks go to my doctoral advisor Prof. Dr.-Ing. Udo Lindemann for the intense support of my research and his confidence in my ideas. Especially, his provision of scientific freedom and supporting reviews provided the basis for the successful compilation of the present work.

I want to thank Professor Andrew Kusiak for the acceptance of being the second advisor, his intense review and provision of numerous propositions of improvement. Also, I want to thank Prof. Dr.-Ing. habil. Boris Lohmann for accepting the chair of the examination board and for the organizational handling of the dissertation process.

I want to thank all my colleagues at the Institute of Product Development for the successful and procreative collaboration in projects, research and teaching. Especially the collaboration with Dr.-Ing. Thomas Braun provided closer insights into the requirements of complexity management in product design and largely contributed to my research results. Frank Deubzer and Matthias Kreimeyer helped me with many discussions on the topic and provided important ideas and approaches during several projects.

I want to thank all my student assistants as well as all the students who worked out semester theses and diploma theses under my supervision; the obtained insights and results mainly contributed to my research work. Wieland Biedermann, who participated in my work over a very long time and particularly helped me in various projects and research topics, deserves special mentioning.

Special thanks go to my wife, who always encouraged me in my work and offered patience and comprehension in stressful times. Also, I want to mention my son Yves, who helped me to recognize the important things in life. Finally, I am very grateful to my parents for the extraordinary and permanent support. They offered me the chance for obtaining unforgettable experiences that incorporated important decisions in my life.

Garching, November 2007 Maik Maurer

CONTENTS

1. Introduction

The research presented in the thesis introduces an approach to complexity management that focuses on the connectivity in objects of product development, i.e., the constellations formed by existing linkages. This provides far-reaching possibilities for analysis, control, and optimization of complex products and services in a holistic context, while the amount of data remains manageable. The keywords of this statement require a brief explication:

- Analysis implies exploring the connectivity of an object in order to identify content, and thus to interpret and better understand the considered object.
- Control describes the possibility of interacting with the elements of an object in order to understand consequences resulting from implemented actions.
- Optimization implies adapting the connectivity of an object in order to better suit its desired functionalities.
- The holistic context means to consider all relevant aspects of a complex object and to represent their opposites in order to extract and understand all viewpoints.

The first chapter of the thesis describes the challenge of complexity in product development. Its outline is formed by the following four statements:

- In spite of approaches on avoiding, reducing, and controlling complexity in product development, a steady increase of complexity can be assumed for the past and predicted for the future.
- Whereas different areas of complexity can be identified in the context of product development, the market represents a major source of increasing complexity.
- Structures that result from the connectivity of elements represent an important tool for accessing complexity in product development.
- If complexity can be controlled, it does not necessarily imply negative aspects but can provide competitive advantages in product development.

The objectives of the thesis and the requirements of the solution are based on the initial problem description, which is also clarified in a case study. Next the scientific approach and the background of the thesis is presented. The subsequent classification of the work indicates the embedding in and relation to relevant disciplines, and an overview of the thesis chapters introduces the composition of the thesis.

1.1 Problem description

HUBBERT stated in 2003 that the severe problems with the electronic systems of the Mercedes E-Series resulted from the “complexity of systems” [HUBBERT 2003]. In almost all relevant sections a steady increase of complexity can be observed, for example, because of augmented product functionalities or market diversification. In this context, BULLINGER ET AL. state that an increasing organizational and technical complexity characterizes the product development

in the automotive industry [BULLINGER ET AL. 2003, P. 15FF]. Numerous strategies, like modularization or platform building, are commonly applied in order to minimize manufactured product variants and reduce their internal complexity [BALDWIN & CLARK 2000, P. 5FF]. However, the current methods for managing complexity can result in an inadequate outcome and compromised solutions, and can keep manufacturers from offering customer-oriented and specific solutions to the average customer ([PUHL 1999, P. 20FF], [LINDEMANN 2004B]).

In research work on product development, complexity has already been addressed from many different perspectives ([WEBER 2005], [JOHNSON 2005]). Approaches for its systematic management are provided by the disciplines of system dynamics, operations research, and systems engineering, among others. Despite current methodologies and strategies for better controlling complexity, it still poses an eminent challenge and an important competitive factor for enterprises that use technical products or services [CLARK & FUJIMOTO 1991, P. 129FF].

Increase of complexity in product development

Examples of analyses in the automotive industry of the 1980s already showed high complexity concerning processes and products, and numerous strategies for its systematic management have been designed [CLARK & FUJIMOTO 1991]. Nevertheless, BULLINGER ET AL. mention that the trend towards increasing variant numbers and product complexity will continue in the automotive industry [BULLINGER ET AL. 2003, P. 17]. RENNER describes the increase of available car models at BMW. In 1995, the company offered eleven different car bodies[1] at the German market; this quantity almost doubled in the next ten years [RENNER 2007, P. 19].

Even this example, however, only depicts a small fraction of the rise of complexity at BMW, as the quantity of country-specific variants or selectable-configuration details simultaneously increased as well. Furthermore, a multitude of technical dependencies emerged due to increased integration of mechatronical[2] components in modern automobiles. Consequently, not only product complexity but also process complexity increased, in part because mechatronics implied an intense division of labor and an increased need for development coordination [STEINMEIER 1999, P. 3FF]. Additionally, the increase of product functionalities resulted in the necessity of an intensified cooperation of both internal and external enterprise departments and suppliers [KEIJZER ET AL. 2006]. For this reason, organizational and communication flows also become more complex ([KREIMEYER ET AL. 2006], [MAIER ET AL. 2006]).

Figure 1-1 depicts the four fields of complexity mentioned in product development: market, product, process, and organization. The total impact of all these fields is of major importance; this mutual connectivity is the reason why considering only the isolated aspects of complexity

[1] Four different series with variations like sedan, coupe, and estate car;

[2] "[Mechatronics is…] the synergetic integration of mechanical engineering with electronic and intelligent computer control in the design and manufacturing of industrial products and processes." [HARASHIMA ET AL. 1996]

is often misleading. Such attempts for optimizing[3] complexity only consider system extractions and neglect related aspects with their specific requirements and constraints. Actions derived for improving complexity can consequently contradict requirements that have not been considered. For example, a high product complexity due to a diversified product portfolio could be counteracted by reducing the number of product variants offered. In the extreme case, only one product specification could be provided for the market. Although this would definitely decrease the overall product complexity, the lack of flexibility concerning specific customer requirements could turn out to be problematic. If the market complexity, i.e., the diversity of niche markets, demands high quantities of product variants, manufacturers will lose market shares and sales figures will decrease. Furthermore, avoiding complexity by reducing variants can only result in positive effects if the associated enterprise processes can be simplified in equal measure. The practicability must be assured before noticeable actions are implemented on specific product variants.

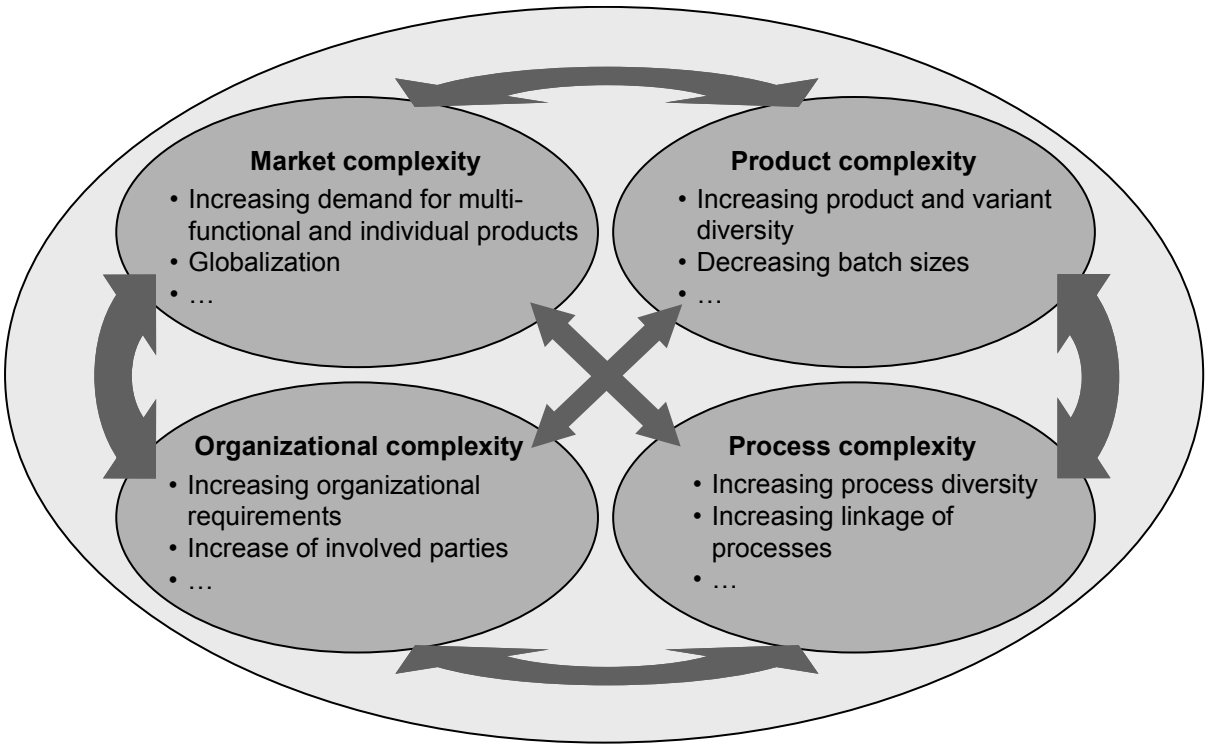


Figure 1-1 Aspects of complexity in product development

Integral modeling or simulation approaches try to avoid the deficiencies that emerge when considering isolated aspects of complexity, but they possess other significant disadvantages, especially in the early phases of product development, as specific product information is often unavailable. Therefore, simulation approaches can not be applied because such applications require abstract descriptions that must be processed by computation. Additionally, the amount

[3] Mostly the objective is the reduction or avoidance of complexity; the here chosen neutral formulation takes into consideration that sometimes even the increase of complexity can be helpful.

of elements and dependencies that have to be considered for holistic models in product development often do not allow for comprehensive and detailed acquisition of all data required to form the modeling basis [STEINMEIER 1999, P. 144FF].

The market as the source of increasing complexity

Generally, new requirements for an enterprise's complexity management can emerge from each of the four aspects shown in Figure 1-1, but the market mostly dominates. On the one hand, market requirements can only partly be controlled by enterprises. Yet, they need to be thoroughly understood in their impact on products. New trends can arise from a multitude of sources that are out of reach for manufacturers and vendors. On the other hand, enterprises will always try to fulfill newly appearing market requirements. Globalization effects are one reason that has been identified as the driving force for enhanced market complexity [CLARK & FUJIMOTO 1991, P. 57FF]. Improved possibilities of communication, more available offerings from different market participants, and worldwide distribution structures result in the opening of new markets and consequently in multiple product variants, such as required country specifications. Also, the market pressure increases as customers possess improved opportunities to choose between large numbers of providers. The tendency towards a strengthened buyers' market results in more requests for product individuality, which causes more product variants, decreasing quantities per variant, and finally increasing complexity, as a challenge for the manufacturer [PINE 1993, P. 33FF].

As mentioned before, a steady increase of complexity in industry was observed in the past. The statements above on market development indicate that this trend will not only remain but actually accelerate in the future. The business strategy of mass customization meets the concerns of increasing product customization by proposing possibilities for the systematic combination of advantages from mass production and efficient product customization [PINE 1993, P. 131FF]. This approach asks for considerable flexibility of product and process development, as well as organizational design. The best possible product adaptability is not only required during the initial product planning, but newly arising customer requirements also initiate recurring changes of existing products. This can cause enormous problems in product development because of dependencies within included components. In order to assure market success, the realization of specific customer requests, i.e., the product changes resulting from them, needs to be identified and rated according to required resources and costs. This demands a complete knowledge of the connectivity of product elements. In other words, the structural constellation of dependencies between product elements is required, but is mostly unknown. This explains why documented case studies on successful mass customization are generally limited to non-technical products, whose components do not possess or only slightly possess mutual dependencies [PILLER & STOTKO 2003, P. 276FF]. For example, clothes can successfully be customized today, as the change of individual measures does not significantly influence further product attributes, components, or the production process. In the case of technical products, e.g., automotives, dependencies within components, processes, or involved people, are extensive and product adaptations require detailed consideration of probable change propagations resulting from the connectivity of elements. A methodical support for these requirements for the application of mass customization to technical products is not apparent so far and is provided by the approach of the present thesis.

ANDERSON describes the trend towards increasing relevance of niche markets and the demand for product variants as the "long-tail phenomenon", which calls into question the current rules of concentration on top-selling products [ANDERSON 2006]. The essential statement derived from this phenomenon is that in many business branches no longer 20% of the product variants produce 80% of an enterprise's turnover[4]. ANDERSON concludes that the strategy of focusing on a small amount of (most sold) product variants will no longer be the most successful one in the future, when the total amount of rarely sold products gains more importance. In fact, the product quantity sold in niche markets becomes more and more important, and has to be met effectively by appropriate product development. Figure 1-2 shows the three main effects responsible for pushing sales figures towards the increasing relevance of niche products. The area shaded in grey describes qualitatively that part of the product variants which is usually considered as profitable due to the Pareto Principle. ANDERSON describes different mechanisms of production and distribution that result in "fattening" the partition of product variants with small individual sales figures. The more these mechanisms occur in practice, the less this product variety of the "long-tail" can be neglected by enterprises. ANDERSON speaks of "democratized tools of production", "democratized tools of distribution", and "connect supply and demand" [ANDERSON 2006, P. 54FF].

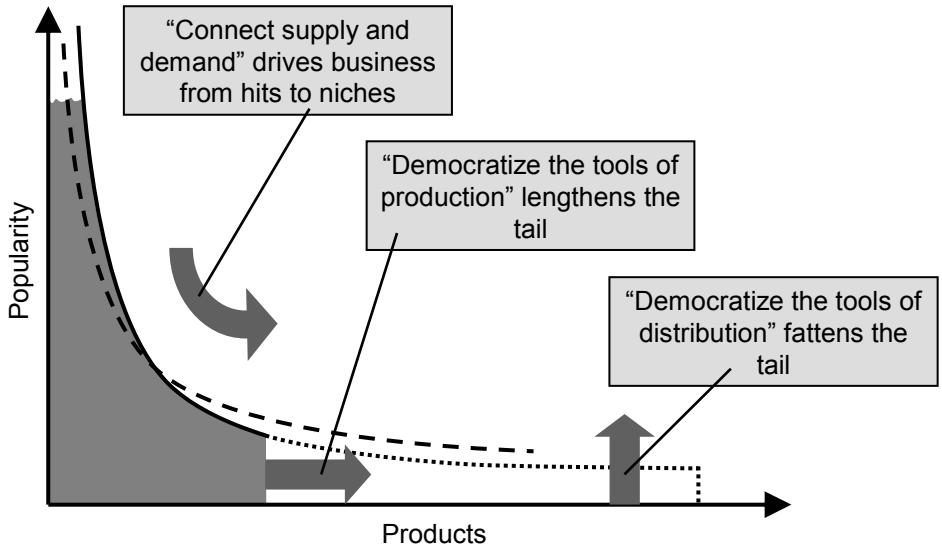


Figure 1-2 Forces of increasing product variety (according to [ANDERSON 2006, P. 54FF])

Nowadays, the profitable focusing on niche markets can primarily be seen with digital products, e.g., music. Further case studies are related to physical products that can at least be ordered and administered digitally, e.g., books [ANDERSON 2006, P. 89]. The reasons for focusing on certain markets therefore are identical with those of the business strategy of mass customization: a profitable product customization requires frequent and short-term

[4] The declaration that 20 percent of the causes produce 80 percent of the consequences is called the Pareto Principle. A further commonly used term for this phenomenon is the "80-20 rule".

adaptations to products, when large quantities of identical products can no longer be sold at the market. Even if a customer only asks for one single adaptation, this can eventually cause considerable change propagation throughout the entire product, if its components are mutually interrelated. If a customer asks for another radio, for example, a number of adaptations may be necessary to the electric system of an automobile. In the case of digital products like songs those dependencies do not exist. Customization measures are restricted to the compilation of songs according to customer liking. The choice of one specific song does not influence further ones, and therefore the "functionality" of the entire product (accessibility of all selected songs) remains unmodified. In the case of technically demanding products, multiple dependencies exist between integrated components and considerably complicate the individual design of a specific component. The determination of consequences that can result from single adaptations requires information about the commonly unknown or not transparent product architecture[5]. Knowledge about this structure of connectivity would permit developers to identify required actions that are related to a specific customer request. Also estimates of resulting efforts and costs would become possible. Ideally, the dependencies between product elements are designed to meet future and so far unknown adaptations; i.e., due to the structure of connectivity, customer requests can be implemented by causing the least possible change impact (in best the case comparable to music). The customization of technically demanding products will be a key factor for future market success and requires adequate methodical support, which enables product developers to analyze, control, and optimize the connectivity of elements.

The role of structure in evaluation of complex systems

Concerning artifacts[6] in general, RIEDL states that their functionality and implied structures are intimately related. He explains that functionality arises from the existence of structures, and conversely, that structures always possess functionality [RIEDL 2000, P. 8]. Furthermore, RIEDL mentions that phenomena of artifacts are typically discovered by existing structures, which are applied for interpretation and serve for explaining the functionalities. Whereas structures can be directly observed when considering an artifact, the explication of functions requires a theoretical construction [RIEDL 2000, P. 8]. As the structure is responsible for many product attributes and even specific system behavior, it is surprising that the relevance of structure consideration for the complexity management is not supported sufficiently by state-of-the-art approaches in product development. In fact, every system[7] comprising at least some parts that interact with each other possesses an underlying structure [BOARDMAN & SAUSER

[5] The term "product architecture" often refers to hierarchical decompositions. As here general constellations of linkages between product elements will be considered, the term "product structure" will be applied. That means that a product architecture represents a specific form of a product structure. A definition of the term structure can be seen in Chapter 2.1.1.

[6] An artifact describes a product or phenomenon that results from human or technical impact.

[7] A system is composed of compatible and interrelated parts that form a system structure, possess individual properties, and contribute to fulfill the system purpose. A closer consideration of the term "system" is presented in Chapter 2.1.1.

2006]. Knowledge about this structure allows for conclusions about the system itself and results in improved understanding. A striking example for this statement is depicted in Figure 1-3, which visualizes a specific system view on the 2006 Soccer World Cup quarter final between Brazil and France. France won the match and Brazil played unexpectedly poorly. As a soccer team is composed of several individuals that interact with each other, the statement about the significance of structures is applicable to this system. If the system behavior is influenced by the structure, the differences in team play should be detectable. Indeed, comprehensive statistical analysis of the match should allow specialists to draw the correct conclusions, but considerations of the structure follow a different approach. Figure 1-3 depicts the passes played successfully between players of both teams. Cases of less than five passes between two players were neglected for the representation. The two diagrams shown are based on strength-based graphs that align elements to the following rule: elements (the players) generally reject each other and get pulled together by existing dependencies. The more successful passes played between two players, the stronger the attraction force of the depicted dependency becomes and the closer the two players are located to each other. The dependencies represent arrows indicating the direction of passes. Arrows in dark gray indicate that passes were played in both directions.

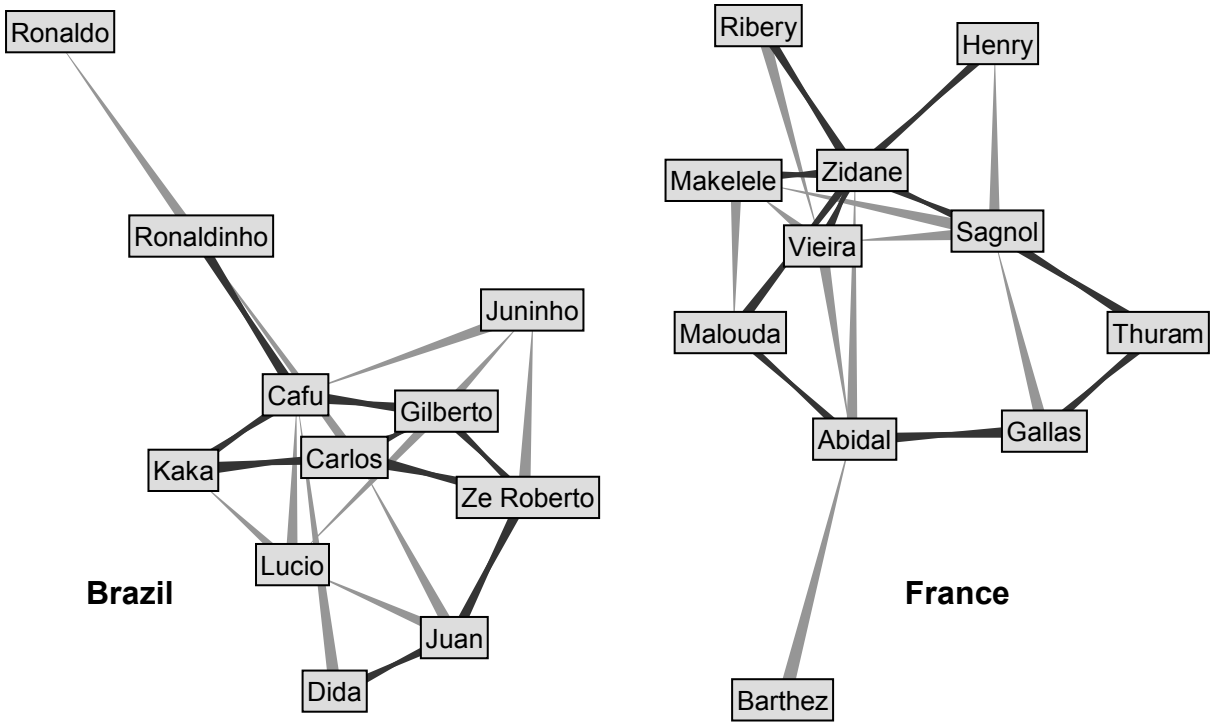


Figure 1-3 Successful passes between players in the 2006 Soccer World Cup quarter final Brazil-France

After looking at the figure, the findings are as follows. When looking at Brazil, it is clear that only one of the attackers (Ronaldinho) received passes from two other players (Cafu and Carlos). Ronaldo only obtained the ball by Ronaldinho and never played a significant number of successful passes to further team members. Hence, Brazil's playing does not seem to be creative from a structural point of view – a statement that corresponds to the real match. In

contrast, the structure of successful passes in the French team turns out to be more variable. Like Cafu on the Brazilian team, Zidane represents the role of a central key player. He played successful passes in both directions with six different team members, including both attackers. The defense of the French team (comprising four players) is connected by bi-directional successful passes, which indicates a well considered tactical play. The goal keeper Barthez is only minimally related to the other team players (in contrast to his counterpart Dida in the Brazilian team), which indicates his non-integration in most of the defense work. Of course, several other structural aspects could be identified, but the significance of the example becomes clear: the consideration of system structures allows closer insight into their behavior and provides the basis for meaningful analysis and interpretation. Transferred to technical systems, it is surprising that the design of structures generally represents a passive process. Every product development necessarily results in a product structure, but its active consideration has been limited to only a few applications so far, e.g., to the application of a modularization strategy. A methodology focusing on the consideration of structures in product development seems to be promising as an approach for enhancing the possibilities of the analysis, control, and optimization of complex design.

Opportunities due to complexity in product development

Regarding strategies for avoiding or minimizing complexity, one can assume that complexity must be prevented by any means. However, complexity does not represent axiomatically negative characteristics in product development. The enhancement of complexity may also allow more flexibility in product design; if, for example, the implied complexity refers to the quantity of product variants offered, an increased product variety can better match different customer requests that arise [SCHUH & SCHWENK 2001, P. 61FF]. This demands effective possibilities for controlling this kind of complexity that enable enterprises to benefit from a wider range of products offered. For this reason, the present thesis is not only focused on complexity reduction, but aims at the creation of competitive advantages due to the control of structural complexity.

Complexity can be an initial barrier against product plagiarism [WILDEMANN ET AL. 2007, P. 50FF]. Besides taking legal steps, this plagiarism is often avoided by raising the technological barrier; thus, protection against counterfeiting often depends on how technologically advanced a company is. As standards become more uniform in a globalized economy, the increase of product plagiarism seems to be a logic consequence even in complex products. However, if enterprises are able to manage the complex structural dependencies of their products, they can quickly identify potentials and risks of product adaptations and create suitable actions. In the case of a distinct product change, this allows early cost estimations as well as estimations of the required resources and know-how, as all adaptations resulting from the initial change are on hand. Therefore, the suitable management of structural dependencies helps optimize the entire process of adaptation design. Market participants trying to counterfeit products do not inherit the entire structural knowledge and consequently are handicapped concerning the adaptation process. Product counterfeiting requires a new variant of the original product that is made available at the market. If the

original manufacturer can reduce the duration of product model cycles[8] and can diversify the quantity of products sold for a wider range of product variants, product plagiarism becomes increasingly difficult and unprofitable [WILDEMANN ET AL. 2007, P. 49F]. This trend is described by the long-tail phenomenon mentioned before [ANDERSON 2006]. The comprehensive control of complex structural dependencies in product development can permit the realization of such a strategy for technical products in the future.

An important aspect of preventing product plagiarism is that the knowledge about structural dependencies within a product spectrum[9] does not represent an integral part of the products that leave the producing company [LINDEMANN & MAURER 2006]. Typically, potential product falsifiers re-engineer products appearing on the market in order to obtain the product know-how. However, the structural dependencies of the development of a product spectrum can only be extracted insufficiently; if a complex product is meant to be customized in the future, actively developed dependencies are designed to serve a multitude of possible variant specifications. However, only single specifications are available on the market that do not allow extracting the comprehensive design. Only the knowledge of the different specifications would permit an increased understanding of the structural composition [LINDEMANN & MAURER 2006].

Increasing product diversification becomes extremely important in many industrial branches. Modern business strategies, like mass customization and focusing on the benefits from the long-tail phenomenon, show the principal possibilities of its application. The consideration of product structures for the analysis of system behavior, and the control of complex interrelations and their structural optimization, form a promising methodological approach that has not been realized so far for application to product development.

1.2 Case study: Race car development

The case study considers the project TUfast[10] located at the Technische Universität München (Germany). The objective is a race car development that complies with the formula student regulations issued by the SAE International[11]. Besides this case study, the author further executed several other projects on complexity management in product development, but data and findings are subject to nondisclosure. Although data anonymity would allow the publication of extracts, such examples would reduce comprehension of the thesis' approach. Information on the product development of the project TUfast could be captured without severe constraints for its publication. Ascertainable data are of high quality, because

[8] The product model cycle describes the period of time between the releases of new product versions.

[9] A product spectrum describes the total amount of product variants that can be produced, i.e. the product spectrum comprises of all possible degrees of freedom of a product [LINDEMANN 2004A, P. 499]. A product spectrum can contain free spaces that allow for improved specifications of variants, as variability is already pre-planned [LINDEMANN & MAURER 2006].

[10] TUfast - Student Racing Team: http://www.tufast.de

[11] SAE International, description of competition rules: http://students.sae.org/competitions/formulaseries/rules/

information acquisition was executed iteratively with the possibility of detailed inquiry. As the formula student competition takes place once a year, it was possible to attend all steps of the development process. Finally, the findings of this thesis could be fed back into the race car development in order to validate their significance.

1.2.1 Project description

The project TUfast is an initiative to develop race cars compliant to the international Formula Student rules. The team comprises of approximately 25 members; most of them are students of mechanical engineering. Due to yearly competitions with adapted rules each time, the overall time of the development process is limited to around ten months. The competitive criteria are not limited to the results of the running of one single race, but extend to the entire development as well as race results in different disciplines: engineering, documentation, and design to cost are evaluated, as well as handling, performance, durability, and fuel consumption. The maximum budget is fixed by the rules of the Formula Student. The accordance with these rules has to be documented in a cost report that makes up part of the competition. Figure 1-4 depicts the Digital Mock-Up (DMU) model of the race car (version nb06), realized by use of Catia V5[12] software. The car possesses a carbon monocoque and is powered by a 600 ccm motorbike engine that provides 86 PS. Many components, from the drive to the suspension, are completely produced in-house or are adapted for the application scenario. This results in a compact design and a total weight of less than 200 kg.

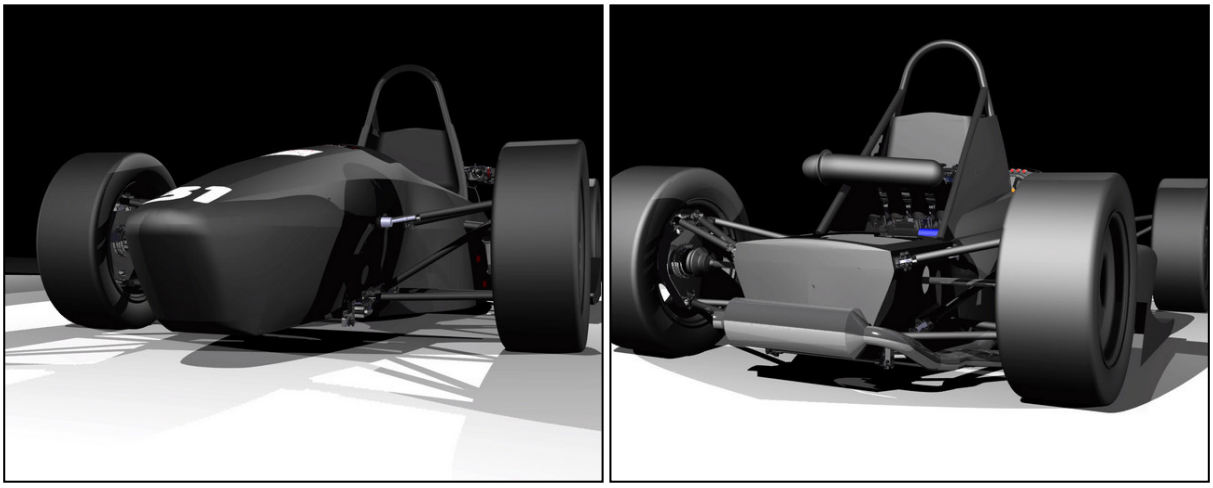


Figure 1-4 CAD model of the TUfast race car (version nb06) [TUFAST INTERNET PAGE 2007]

With an overall size of 25 members, the team requires intensive coordination. The organizational design is historically grown and does not necessarily correspond to all actual requirements. As the TUfast group is a student initiative, the team members are not occupied full-time with the project and their available time varies due to other university activities.

[12] Catia V5, Dassault Systemes: http://www.3ds.com/products-solutions/plm-solutions/catia/overview/

The Bill of Materials of the race car comprises 130 components. However, components are kept on a general level of detail, e.g., the engine is listed as one single component. The compliance to the rules as well as assurance of competitiveness on the track requires adaptations of imported components for the specific requirements (e.g., a mountain-bike brake system has been implemented for the race car). In total, imported parts are purchased from 28 different suppliers; besides the technical complexity of adapting the components, the complexity of the process is also increased, as many of these parts are produced only on request and possess long delivery times.

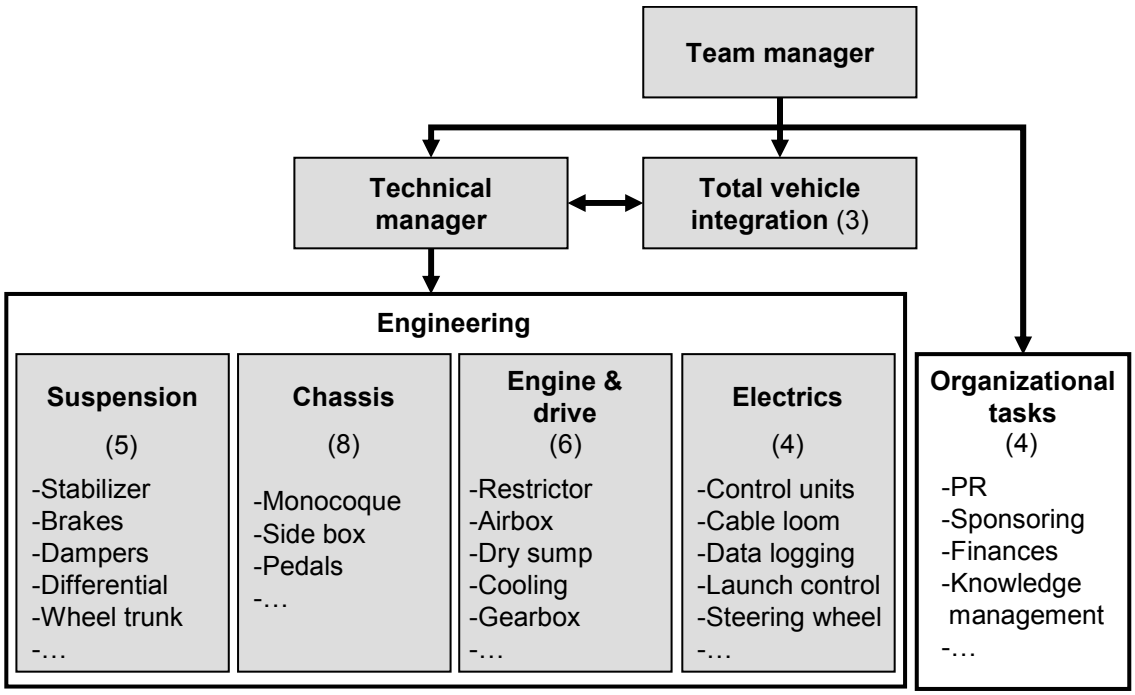


Figure 1-5 Organizational structure of the project TUfast

The organizational design of the project TUfast is depicted in Figure 1-5. The classification of engineering disciplines is established according to the functional units of the car. The group that is responsible for organizational tasks is assembled as well. Figure 1-5 lists in brackets the amount of people engaged per group. In each engineering section one technical manager is responsible for the coordination, and one superordinated technical manager presides over the entire engineering development. Paralleling the technical manager, a group for the "total vehicle integration" is implemented. This group coordinates the compatibility of components, the identification of emerging conflicts, and the validation of the constructive integrity by virtual assembling in the CAD software. In practice, some people are assigned to two different units or act as designer and technical manager at the same time.

Commonly, team members work two days per week on the project. Less working time is available during Christmas vacation (mid December to mid January) as well as during the exams (February to March). Typically, students stay in the team for two years, i.e., they take

part in two entire development cycles. Consequently, approximately half of the members quit the team each year and have to be replaced by newly recruited students.

Figure 1-6 shows the project plan with the essential milestones. Initially, the team organizes a concept workshop for determining the general development. Subsequently, the component design starts, which is terminated by the package freeze in December (final specification of packet interfaces and functionality) and the design freeze in January (final specification of the component design). The production starts simultaneously with the component design and lasts until the milestone "Rolling chassis" in May. At this time, the race car is available in a roadworthy status. By use of "Hardware in the loop"[13] specific components are already tested during production. Some components (e.g., the engine) are tested by implementation in an experimental vehicle (typically the last year's race car). As the competition starts in July, this date is the definitive end of the test phase. The milestone "Rollout" does not present a technical but an advertising milestone, where the roadworthy car is presented to the public.

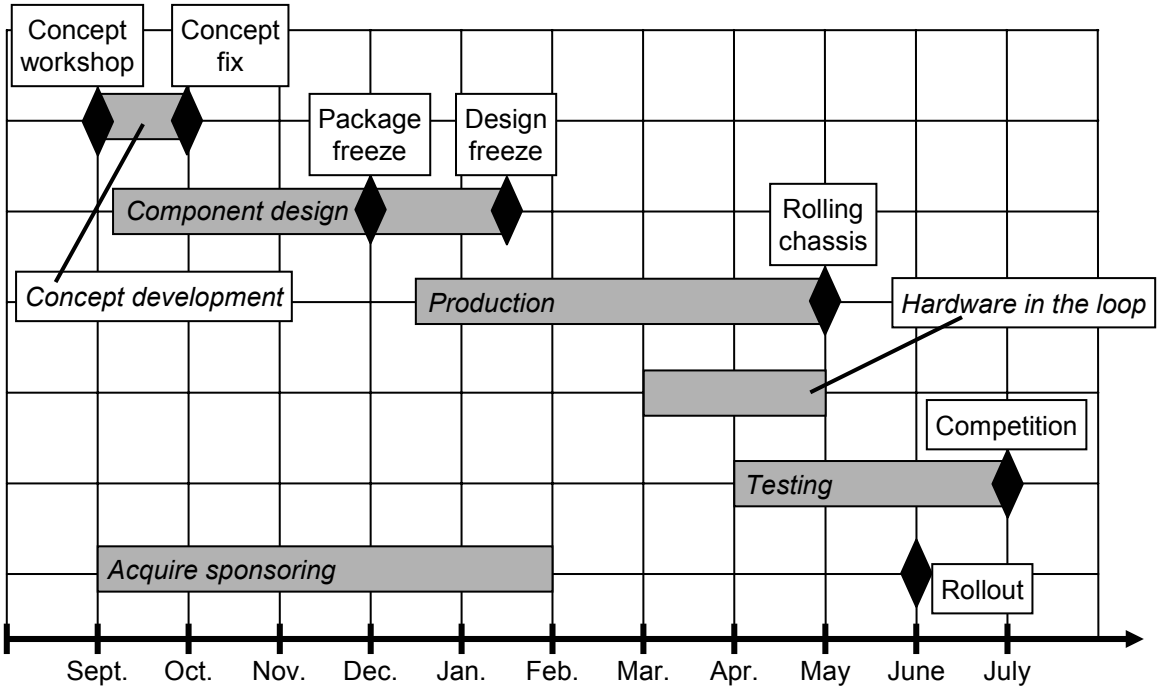


Figure 1-6 Project plan of the TUfast race car development

It can be seen from the project plan that the periods of decreased availability of team members are located at an unfavorable time at the end of the component design and the end of production. This often results in severe discrepancies of the project flow from the theoretical plan.

[13] "Hardware in the loop" is a method to analyze product attributes and characteristics of mechanical or electronic components by their embedding in a real-time simulation [LINDEMANN 2007, P. 271]

1.2.2 Problem description

The aspects of complexity that have been described in Section 1.1 can also be identified in the project TUfast (Figure 1-7). The market with its requirements is represented by the organization SAE International. The competition regulations comprise 150 pages and are affected by yearly changes. Even during an ongoing development period, some rules can be adapted, if design trends enforce security improvements. For example, a trend towards light-weight cars with minimum frame constructions resulted in quickly changing the rules for the dimensions of the frame geometry in order to assure better protection of the pilots. Therefore, a ongoing survey of the conformity to the actual rules (the market) is mandatory. Like a typical market situation, the project team has to consider many requirements they can hardly influence and which may be prone to rapid changes. Several products (here the race cars) are offered to the market and only the best succeed.

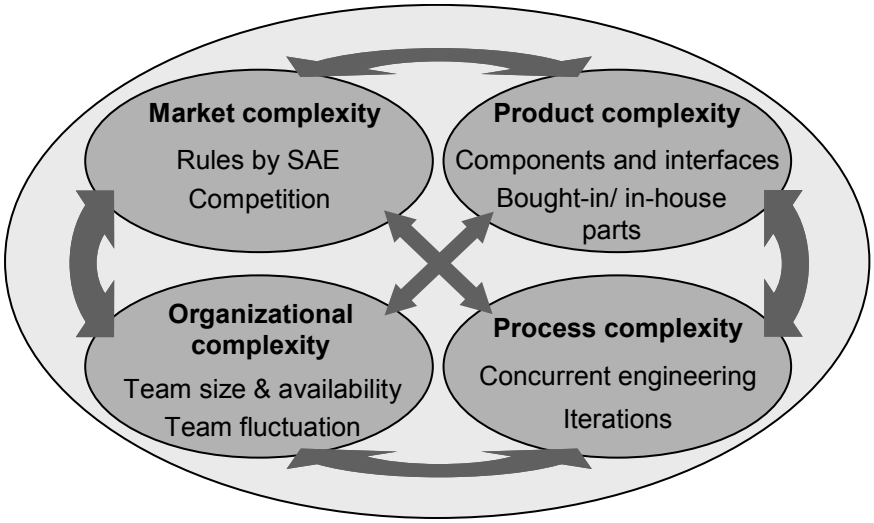


Figure 1-7 Complexity in the TUfast race car development

The project represents a prototypic development and is characterized by knowledge deficiencies, e.g., the attributes of imported components or the production of in-house components. This means significant product complexity. The development process is characterized by complexity, as the short development time requires concurrent engineering and the prototypic development produces much iteration. The different process steps are interconnected and a single adaptation affects several other process steps. An organizational complexity, for example, results from the team fluctuation. As approximately 50 % of the team members change every year, knowledge transfer and reorganization of the team structure pose major issues.

Modification of product components

In terms of product complexity, the concept of the 2006 race car was to displace the brakes from their positioning beside the wheels to the middle of the car. Unfortunately, the required imported parts were not available in time. This resulted in an unexpected adaptation late in the

development process, as the entire brake concept had to be changed (see Figure 1-8). A significant amount of change propagations had to be managed, and only some of them were directly obvious to the designers (brake disks and adapters, brake calipers, etc.). Some required adaptations only became visible slowly, when the designers assembled the changed brake system. This caused several time-consuming iterations before the new concept was implemented successfully.

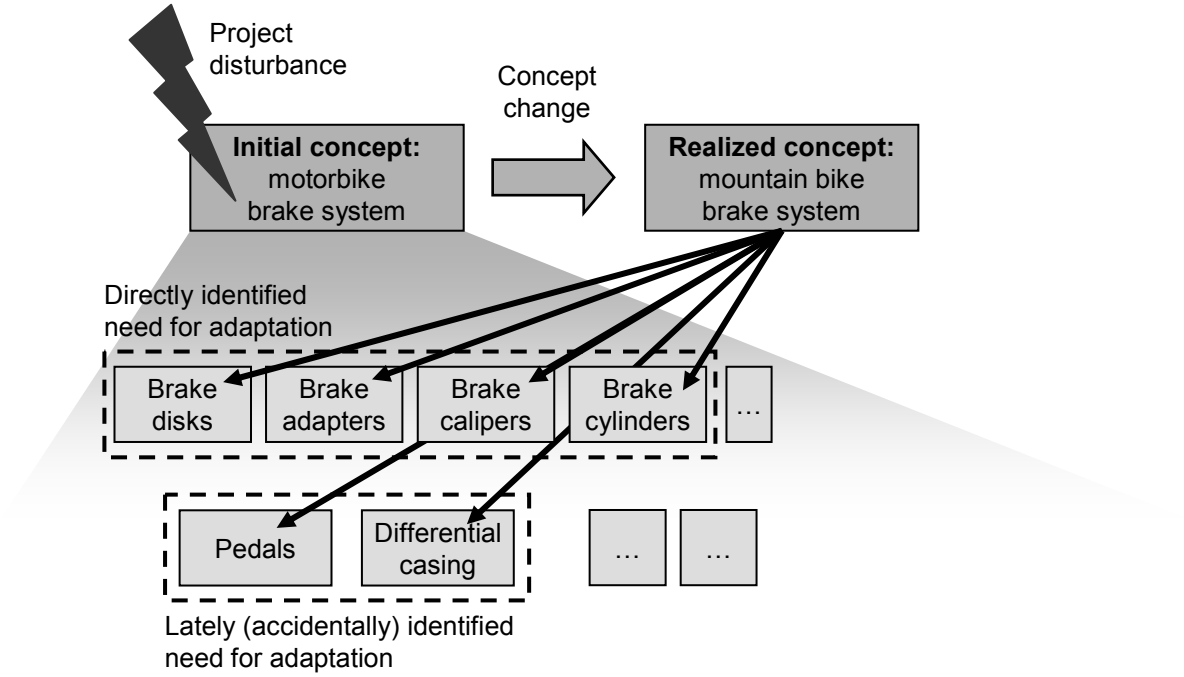


Figure 1-8 Change of the brake concept and resulting component adaptations

Organizational change

During the component design the team member responsible for the engine control unit left the team. In the beginning, this did not appear to endanger the project success, as another person with the required technical know-how was quickly found. Although the development of the engine control unit worked fine, the team management did not consider the required information flow. The former developer was well integrated in the team through long-term personal relationships with most of the other team-members; information exchange was implicitly organized. The newly acquired developer was not aware of the other team members who required information on his specifications. Whereas for the electrics engineering team communication still worked well, engine engineering and suspension engineering teams did not receive the newly implemented specifications. The deficiencies only became obvious when assembling the race car, because components did not match (Figure 1-9). The engine and suspension engineering was based on out-dated specifications concerning their interfaces with the engine control unit. This resulted in several design iterations late in the development process and endangered the project's success, as a roadworthy race car was only completed shortly before the competition.

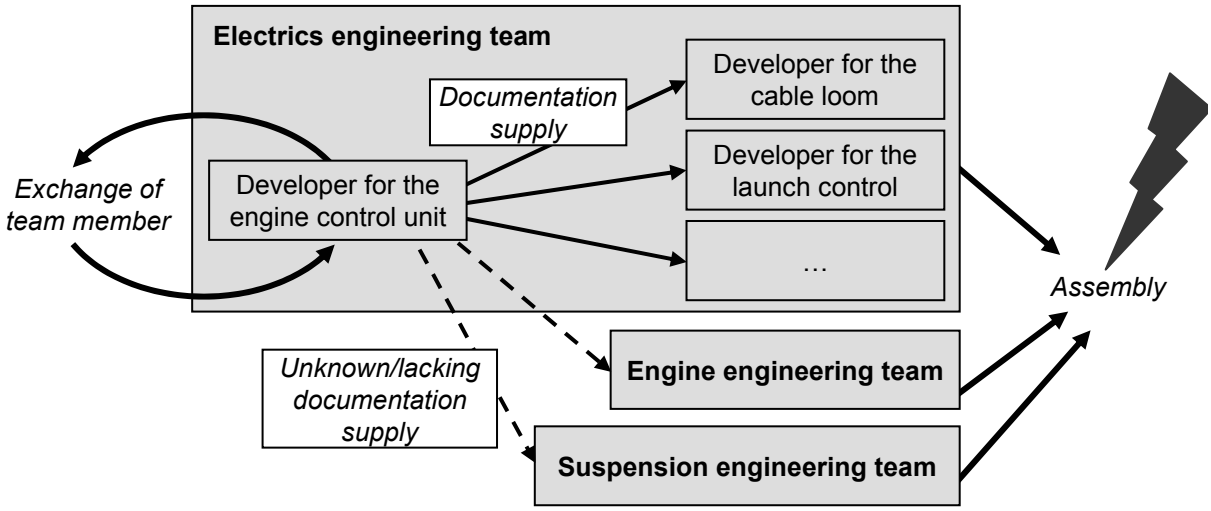


Figure 1-9 An organizational change causes disturbed information flow

Connectivity within the project TUfast

Unexpected adaptations hardly seem to be avoidable and can result in harmful consequences due to the connectivity in the development project. For example, a small extract of the complex connectivity in the project TUfast is depicted in Figure 1-10. Even if only five components, four people, seven documents, and three process tasks are depicted in their connectivity with some dependency types, the consequences of individual adaptations seem to be impossible to survey. This illustrates the need for an approach to control complexity in any product development that is characterized by multiple aspects.

Opportunities due to improved structural considerations

Knowledge of the structure of dependencies would permit a significant amelioration of project problems. Adaptations would become easier to plan, as products and people related to change propagations could be quickly determined and easier and faster accessibility for better controlling required actions would be available. Furthermore, elements could be rated concerning their risk and potential in case of changes. This characteristic in case of changes describes the suitability of product components, processes, or organizational units for adaptation. A high change potential of specific elements means that adaptations to these elements do not result in a large change propagation and can consequently be implemented with a minimal effort. Product components possessing a high change potential represent preferred candidates for a late specification in the design process. Conversely, components that possess a high change risk should be specified early and adaptations should be avoided at all cost (as a large number of subsequent adaptations can occur). If ratings of change risks and potentials are on hand, the team management could better decide where to enforce innovative (and competitive) designs and where to apply established (and hopefully fail-safe) technology. This would reduce design details that would endanger the project in critical areas of the car concept, and innovative design features could be better integrated due to the

robustness of the adaptations. Nevertheless, if unexpected engineering changes occur, any required actions can be effectively determined by considering the implied structure. Finally, future development projects can best profit from structural considerations, if acquired structural knowledge and findings are implemented in the early concept of subsequent race cars.

The case study showed the need for an approach to complexity management that focuses on structural dependencies and takes different aspects of complexity into account. In Section 4.2 the same case study will be examined again and considered by means of the thesis approach (see Chapter 3).

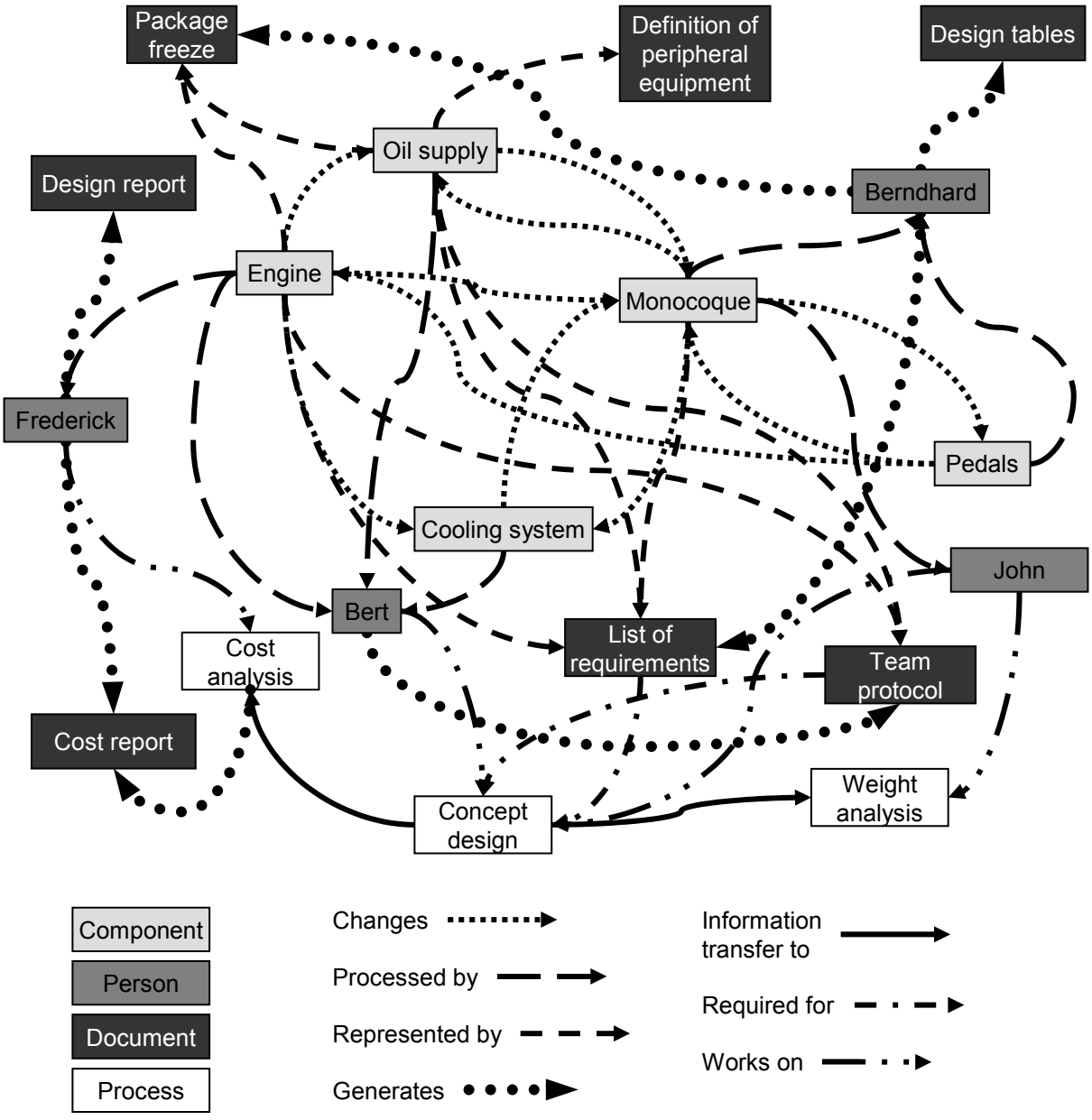


Figure 1-10 Connectivity of system elements in the project TUfast

1.3 Objectives

The overall objective of the present work is to provide a methodical approach to analyze, control, and optimize complexity in product development by focusing on the underlying structural dependencies. This will allow improvement of product development in spite of increasing complexity and enable the implementation of upcoming business strategies that focus on enhanced product customization.

The approach is based on the finding that, on the one hand, each technical product, as well as the processes of its creation, can be accessed by its structure. On the other hand, the structure generation itself generally emerges without being noticed or controlled (passive structure synthesis), and therefore often does not apply to the requirements of the product or process. However, the structure determines the attributes and behavior of products and processes [MAURER ET AL. 2005]. RAPP even found that in some industry branches dominant product structures have been established, which show the potential for superior market success [RAPP 1999, P. 93]. Some product characteristics can even be defined in their structural composition only, for example, the robustness to component changes [COYLE 1996, P. 6]. Some products can be adapted to changing boundary conditions by implementing changes that do not cause further consequences to other product components. Other products need to implement several adaptations or a basic redesign, because a multitude of product components are affected by one initial change. Often components are embedded in characteristic substructures that allow for an early estimation of the system behavior in case of changes. For example, cyclic dependency chains between components can cause self-energizing or self-impeding effects [FORRESTER 1980]. Hierarchical substructures can produce the spreading of changes to numerous product components (avalanche effects) [LINDEMANN ET AL. 2005]. And the creation of clusters based on several components can indicate the potential for modularization [BALDWIN & CLARK 2000, P. 63FF].

Existing approaches of complexity management focus either on isolated aspects, e.g., when optimizing variant numbers from a production view [SCHUH & SCHWENK 2001, P. 71FF], or address a comprehensive modeling approach, which is implemented at an abstract level of detail [STEINMEIER 1999, P. 69F]. The dilemma between the consideration of extracts and global views that are difficult to manage can be solved by applying a structure-based approach of complexity management. Here only a qualitative[14] modeling can be considered, as detailed data that allow for specifying computational relations are normally unavailable. Structure considerations of complex systems must enable users to understand the system behavior as well as the potentials and risks of adaptations by reasonable efforts. Therefore, structure analyses have to point out relevant aspects and starting points for optimal system development. In this context, the term relevant refers to the case-specific needs in a distinct development situation and is meant to limit investigations to only content that is necessary. The systematic modeling of dependency structures must allow the identification of meaningful substructures according to their internal composition and their embedding in

[14] Here the term “qualitative” means that considered data can not be measured by discrete values; the term is applied as the complement to the term “quantitative”.

surrounding structures. If required, they can then be applied to more detailed considerations. The derived objectives of structure considerations for the complexity management are further detailed in the following:

Systematic determination of relevant domains

When starting with a complex problem, it is important to determine which aspects are to be considered in order to answer a specific question. Improper simplification or the extraction of single aspects must be avoided, as the consideration of such partial models eventually hinders the re-transfer of the outcome to the initial problem. Users can draw incorrect conclusions that may result in an unfavorable impact to system domains that had not been considered. In contrast, all aspects must be excluded from considerations that are irrelevant for the specific question. Their inclusion can unnecessarily complicate data acquisition, modeling, analyses, and interpretation. Besides the identification of relevant domains, their modeled level of detail must also be specified. A systematic approach is essential for assuring the quality of results, as the scope chosen provides the basis for all further steps of the approach presented here.

Information acquisition

The acquisition of dependency structures represents a special challenge for the entire concept. Often system structures only exist as implicit knowledge and are shared among several specialists. The multitude of components and possible dependencies in complex systems requires a methodical acquisition that encourages users not to forget or neglect relevant information content. Additionally, data quality and correctness must be assured during the acquisition process, e.g., by appropriate plausibility checks and iterations. The acquisition method must be applicable to team work, and once determined, agreements and decisions concerning structural content have to be clearly documented. Users must always be able to have access to the overview of the system considered and the acquired content, even if significant data exist or if the data acquisition is split up in several workshops.

Condensing information to individual views

Although during the acquisition of structural information all relevant aspects must be included, their simultaneous representation will generally result in a lack of clarity. Users need representations that focus on case-specific aspects, where all relevant dependencies are integrated. This asks for the target-oriented condensing of structural content instead of the extraction of single parts. This condensing must be realized without time-consuming computation and for all user-defined aspects of complex systems. This allows users to obtain a comprehensive understanding of the system’s connectivity.

Selection of relevant system views

Users can obtain comprehensible representations without neglecting relevant aspects by condensing information from specific system views. Unfortunately, complex systems comprise a multitude of aspects and dependency types that allow many system views. Methodical support is required to enable users to systematically select the relevant ones. Furthermore, complex dependency meanings can result from condensing information, as

mentioned before. The relevance of dependency networks for further consideration must be evaluated, as further analysis, interpretation, and optimization efforts have to be kept to a minimum.

Analysis of dependency structures

Dependency structures must be analyzed concerning the existence of characteristic structural attributes that may suggest system behavior. As many characteristics originate directly from a system's structure, executing analyses should provide better understanding of the system. Analysis methods that have already been applied to product development have to be evaluated according to their scope of applicability and eventually have to be amended by approaches from further research disciplines. As complexity also arises from the high quantity of implied elements and dependencies, it must be assured that even extensive structure models can be analytically considered.

Structure manual

Generally, structural considerations should facilitate the users' interaction with complex systems. Decision-making in product development is often complicated by a multitude of dependencies that are only known implicitly. Therefore, the resulting consequences can not be reliably predicted. The approach presented here will allow the creation of a structure manual that points out relevant system elements and structure constellations. The structure manual will help estimate the potentials and risks of component changes by rating the suitability of components for changes. Therefore, users can obtain a more comprehensive understanding of the system and can better plan any required actions within a complex system.

Identification of suitable structural designs

Besides an improved interaction with complex systems, their structural optimization is also enabled. Suitable methods are partly available, but have not been applied systematically to product development thus far. Some structural optimization approaches have became well-known, e.g., modularization and platform strategies in automotive industry. These possibilities must be amended and provided at a generic level for application to user-defined systems. An important use case represents the design for product robustness, representing a product's suitability for adaptation. A high product robustness means that customer-induced product changes can be realized without extensive second-level changes.

Proactive design of appropriate system structures

Whereas structure optimization aims at the improvement of existing products or development processes, the initial development of appropriate structures represents a logical further step. Obviously, proactive and case-specific structure development supersedes any previous structure optimization. In conventional product development, some related methods are available that must be evaluated according to their suitability. The "Design for Assembly" or the "Design for Modularity" is especially well established; however, these methods use only a few structural characteristics and therefore need a systematic enhancement. If product

designers dispose of the possibility of generating appropriate structures, increasing product quality and performance can be generated from design activities.

1.4 Solution requirements

A suitable approach for structure consideration in complex product development must match certain important requirements, which are listed below:

Generic approach on multi-domain modeling

Generally, product development represents an application area that involves different domains, e.g., components, functions, processes, or people. An isolated view of a single domain may be suitable for a specific application, for example, the functional modeling[15] in early development phases [LINDEMANN 2007, P. 117FF]. An examination of domain-spanning dependencies is important in order to obtain an overall understanding. Such a combined consideration of several domains has been realized in methods like Quality Function Deployment (QFD) or Failure Mode and Effect Analysis (FMEA) ([AKAO 1992], [REINHART ET AL. 1996, P. 86FF]); however, these methods represent applications for only partial aspects of product design. A systematic approach for modeling of multi-domain dependencies has still to be realized. The solution desired has to be applicable to all relevant aspects of structure-based complexity.

Intuitive understanding of representations

Structure considerations have to permit a user's intuitive comprehension of the connectivity in complex products, and therefore call for specific requirements for the representation of interrelated data. Different approaches, such as graph trees, lists, or matrices, are available and have to be applied according to specific needs, e.g., during execution of data acquisition or navigation in the structure.

Separation of data processing from data representation

A characteristic of complexity is the large number of elements and dependencies between them (see Section 2.1.1); the interaction with complex systems requires the management of large amounts of data. Here, the representation, on the one hand, and the analytical processing of system information, on the other, have to be separated in order to permit the most effective approach [BROY 1998]. Representation forms have to suit specific user requirements, e.g., intuitive understanding. For this reason, it can not be assumed that all informational content of considered structures is accessible in each representation. Processing information from complex systems often creates different demands than their representation, and therefore must be independent from chosen depictions, such as graphs or matrices.

[15] In comparison to the original, a model represents a target-oriented, simplified formation analogous to the original, which permits drawing conclusions to the original [LINDEMANN 2007, P. 331].

Qualitative and quantitative modeling

It must be taken into account that the application within various (and early) phases in product development requires the possibility of modeling non-quantified information. As complex systems typically comprise many system elements and dependencies, modeling approaches which require detailed specifications or even mathematical descriptions are not suitable, as often the acquisition efforts would become too immense. Furthermore, complex systems are frequently characterized by a lack of knowledge concerning the systems' dependencies. The intended approach has to allow for the acquisition of general structure constellations that serve for identification and interpretation of relevant subsets. These subsets can then be expanded later on by additional descriptions of system elements and their connectivity. Thus, non-quantified information must be the basis of the modeling approach. Nevertheless, it should be possible to implement information on the quantification of system elements and dependencies in case of their availability.

Provision of suitable analysis methods

Generally, numerous analysis methods are available when considering complex system structures. To some degree, these methods possess different analysis objectives, e.g., the characterization of an entire network or the specific embedding of a single component in its environment. These analysis methods also show different suitabilities for specific structural scenarios, e.g., for static networks (representing the connectivity of product components) or dynamic models (representing process flows). In order to permit an adequate selection of useful methods, a classification concerning the analysis objective and scope of application is required (see also Figure 2-8). Analysis methods are often based on specific forms of data provision, such as square matrices or lists. The flexible adaptation of analyses or their combined application can be helpful in many use cases, but requires an integrative description form, for example, one provided by the mathematical graph theory. A common description form permits the transfer of analysis methods from their actual fields of application to other promising ones (e.g., the transfer of methods of process analysis to product structures).

It should be possible to enhance the scope of consideration depending on the specific use case. If, for example, the structural dependencies between system components are on hand, these should be applicable for the enhanced analyses of system functions, organizational responsibility or customer requirements.

1.5 Background and research methods

The basis of the applied scientific approach of the present work is the iterative design research approach according to MINNEMANN ([MINNEMANN 1991, P. 16] and STETTER [STETTER 2000, P. 6]). This approach represents a cyclic process composed of three main steps: observation, analysis, and intervention. The general process is depicted in the center of Figure 1-11. By passing through this cycle, the thesis approaches could increasingly gain relevancy to practice.

The project TUfast represented the primary object for consideration by the iterative design research (see also Section 1.2). To examine in detail the present thesis, the development

process was observed, and the interactions between the project domains involved (e.g., component design and work flow organization) became of major interest. Observations were analyzed and applied to the systematic compilation of the thesis approach. The execution of analyses permitted the identification of deficiencies in the project development process. The analyses generated hypotheses of the thesis based on the analyzed deficiencies and were aimed at the improvement of the development process. In particular, possibilities concerning the mapping between multiple domains were iteratively developed and implemented in the development process of the project TUfast. This represented the intervention phase of the iterative design research approach and was followed by observing the resulting changes to the development process.

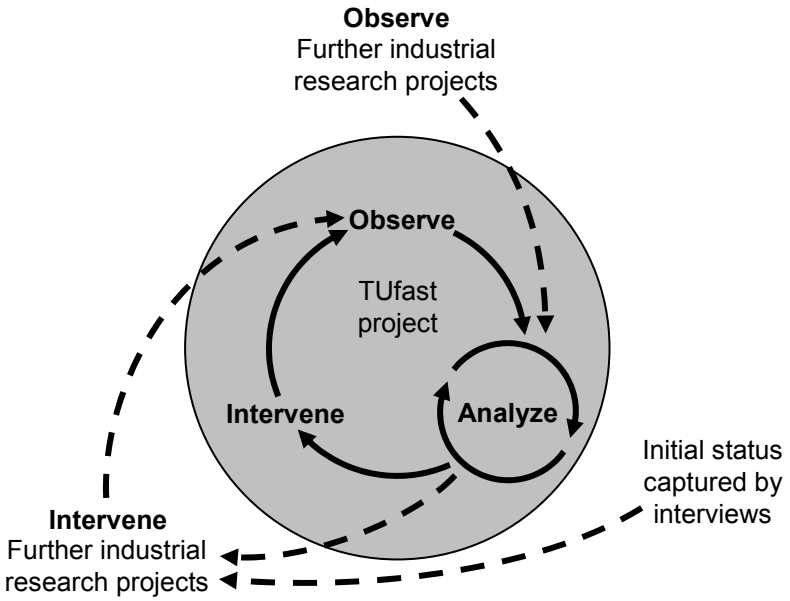


Figure 1-11 Iterative design research approach by consideration of multiple projects

In addition, the iterative design research approach was enhanced by considering multiple projects, as can be seen outside of the circle in Figure 1-11. Besides the TUfast project, further industrial research projects were observed, which permitted the identification of deficiencies in the applied development processes. Based on this, hypotheses were created to obtain significant process improvements. The initial status of the industrial research projects was captured by interviews, which helped validate the resulting thesis approaches. That means that comparative analyses were executed instead of initial observations of the industrial research projects. The chosen scientific approach largely corresponds to the "Design Research Methodology Framework", which is proposed by BLESSING ET AL., but, in particular, meets the concerns about direct applicability to practice because of the particular focus on short iterations [BLESSING ET AL. 1998, P. 44].

The compilation of the present thesis is based on the experiences and findings the author achieved during his research work at the Institute of Product Development at the Technische Universität München. The author worked in the collaborative research center SFB 582, named "Marktnahe Produktion individualisierter Produkte" (Production of individualized products

close to the market) [LINDEMANN 2004A, P. 141FF] and was engaged in the sub-project "Strukturplanung für individualisierte Produkte" (Structure Planning for Individualized Products) [LINDEMANN 2004B] for two years. The objectives and findings of this research project represent the fundamentals of the thesis. The existence of complex dependencies in between elements of comprehensive product spectra, in particular, pointed out the need for improved methods to manage structural complexity in product development [LINDEMANN & MAURER 2006]. Early use cases and exemplary system structures were derived from this project for validation purposes.

The author worked on several research projects in cooperation with industry. Through an agreement with Audi AG, the author attended a research project on knowledge management in the design process. In cooperation with two external suppliers, ontology-based techniques were applied to the specific use case [PONN & LINDEMANN 2006]. The author gained further insight into the complex requirements of information processing in the automotive industry and actual constraints on information access for designers. The project outcome contained a prototypic implementation of a knowledge tool permitting designers to obtain information already available, even if the applied vocabulary in queries varied. The results showed the importance of suitable user access to complex systems. The author profited by his comprehensive experience of dependencies in development processes from an informational perspective.

Several projects on the structure analysis of complex products with small or mid-sized companies allowed the author to obtain extensive experience with the industrial practice of complexity management in design. During these projects, he was able to observe applied tools for system modeling and analysis and rate them due to their usability. Case-specific requirements for controlling complex situations were identified and introduced into the present thesis.

The author's participation in research work on methodologically finding solutions for crisis situations in the automotive industry provided insight into specific requirements for finding solutions when under certain time constraints. A methodological concept was designed that allowed companies systematic fault diagnostics in order to minimize design iteration loops in crisis situations [DICK & LINDEMANN 2006]. The aspects of change propagation in complex design were of particular importance for this project and are taken up in the analysis strategies of the present thesis.

During industry collaboration with the mid-size company Electrostar, the author explored the possibilities for the analysis and visualization of complex structures of market dependencies [BAUMBERGER ET AL. 2006]. The project results validated basic analysis methods and their prototypic implementation. Discussions of the results provide further requirements for structure analyses and determine further research questions discussed in this thesis.

A seminar held with Audi AG provided the author a close insight into modular design in the automotive industry. The project objective "Concept design of a module-based door opener" was executed during a six-month period with a development team of eleven students and two scientific assistants. The early phases of the modularized product development were particularly addressed in the project. Due to the development objective as well as the large

team size, complex dependencies between different domains were especially relevant for this project.

Concerning the relevance of complexity management in product development, the author has co-founded the research group Plegmatum, located at the Institute of Product Development. "The aim of the research team Plegmatum […] is to gain and mediate areas of competence with regard to the controlling of complexity in product development. Furthermore, increased international scientific exchange, as well as cooperation between industries, are central goals for Plegmatum" [PLEGMATUM INTERNET PAGE 2007].

1.6 Thematic classification of the thesis

The approach presented in this thesis is characterized by the combination of methods and techniques from different scientific disciplines. Whereas the state of the art of applied possibilities is considered in Chapter 2, Figure 1-12 shows the aspects in their mutual classification, which is detailed below.

Fundamentally, the problem description of the thesis can be assigned to the disciplines of design methodology and systems engineering. In reference to the topic addressed, it is not reasonable to clearly separate these disciplines, as similar aspects are addressed in both of them, particularly the aspects of holistic problem solving, multi-disciplinary approaches, and systematic procedures in product development.

The approach presented uses research disciplines and methodologies that have already been applied to systematic product development, but also clearly exceed the problem fields addressed here: possibilities of the graph theory as a section of mathematics goes beyond the scope of its selective application in design methodology; for this reason it only partially overlaps with the field of systems engineering and design methodology in Figure 1-12. Graph theory is applied to the thesis approach for the purposes of data analysis and optimization; basically, numerous problems in product development can be formulated as graph theory problems (as discussed in Section 3.4). Furthermore, techniques of graph theory are also applied for visualizing networked information.

The methodology of system dynamics also serves for information visualization in the thesis approach. Some possibilities of system dynamics are already applied in design methodology, for example, by visual impact networks [LINDEMANN 2007, P. 72FF]. For the acquisition of minimally specified system dependencies, in particular, system dynamics provides comprehensive possibilities and a generally standardized grammar for representations [COYLE 1996, P. 18FF].

In addition to the various options provided by the graph theory, the approach presented uses data mining for structure analyses, interactions with complex structures, and further suggestions for optimization. This relates to the identification of characteristic patterns, which are difficult to detect when embedded in the context of larger system structures [KUSIAK 2000, P. 498FF]. Data mining provides possibilities for the effective consideration of large amounts of data and can therefore be useful for the present work. However, the approach presented only concerns data-mining concepts that focus on interrelated data.

An important step of the thesis approach refers to the transfer of real problems to abstract description forms, which allow the application of structural analyses and optimization. Methodical possibilities are taken from the discipline of operations research, whose applications are also used in product development [WINSTON 2004, P. 413FF]. For example DAENZER & HUBER explain the Critical Path Method (CPM) as a commonly applied method of network planning in systems engineering [DAENZER & HUBER 1999, P. 452FF].

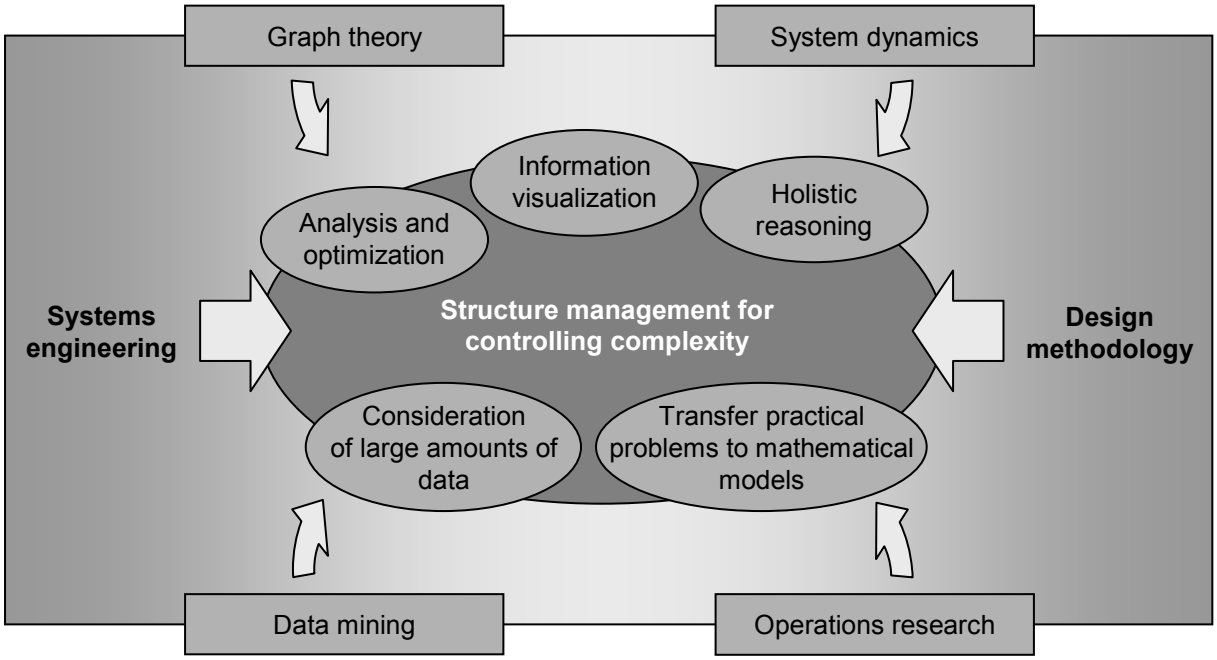


Figure 1-12 Thematic classification of the thesis

In general, the methods and techniques previously mentioned have merged with practical applications, and it is often impossible to define a clear separation between them. For example, graph theory is used in system dynamics as the basis for structure characterizations, as shown by FORRESTER [FORRESTER 1980]. FOULDS describes the use of graph theory in operations research [FOULDS 1992, P. 225FF]. The fact that systems engineering and design methodology are not clearly separated in Figure 1-12 takes into account that both methodologies are closely related by their objectives and often make use of similar approaches. The scheme of classification that is presented here serves to limit the scope of considered approaches.

1.7 Structure of the thesis

The thesis is organized according to the objectives presented in Section 1.3. The structure of the chapters is shown in Figure 1-13.

Chapter 1 introduces the general problem and presents a case study that clarifies the issue of structural complexity. Based on this, the objectives of the present thesis are derived and

solution requirements are specified. Next the scientific approach is presented and the topic considered is classified in reference to adjacent scientific approaches.

Chapter 2 defines the use of relevant terms and explains the fundamentals of the present thesis. Areas of structure consideration and methodologies are considered, which provide input for the approach presented and clarifies the need for the thesis approach.

In Chapter 3 the thesis approach is detailed and an overview of the general procedure is provided. The process steps implied are presented as follows. The initial task is to define the system in question; therefore, the required system aspects and dependency types between them must be determined. Once the system layout is on hand, information acquisition serves to capture relevant structural content. This information has to be represented by appropriate methods and techniques in order to allow user interaction, subsequent analyses, and optimization. Structure analyses apply basic criteria for characterizing complex structures, which are mostly derived from graph theory. The discussion of practices provides procedures that allow for case-specific structural adaptations for optimizing considered structures to be determined. Additionally, a comprehensive collection of basic analysis methods and strategies are mentioned in the Appendix in the form of an easily accessible reference book.

Chapter 4 validates the approach presented, and the initial case study (see Section 1.2) is continued. This allows the reader to perceive the added value for practical problems in the development of technically demanding products. Furthermore, specific aspects of the approach are presented using structures from two other industrial use cases.

Chapter 5 resumes the findings of the thesis and describes possible starting points for future research.

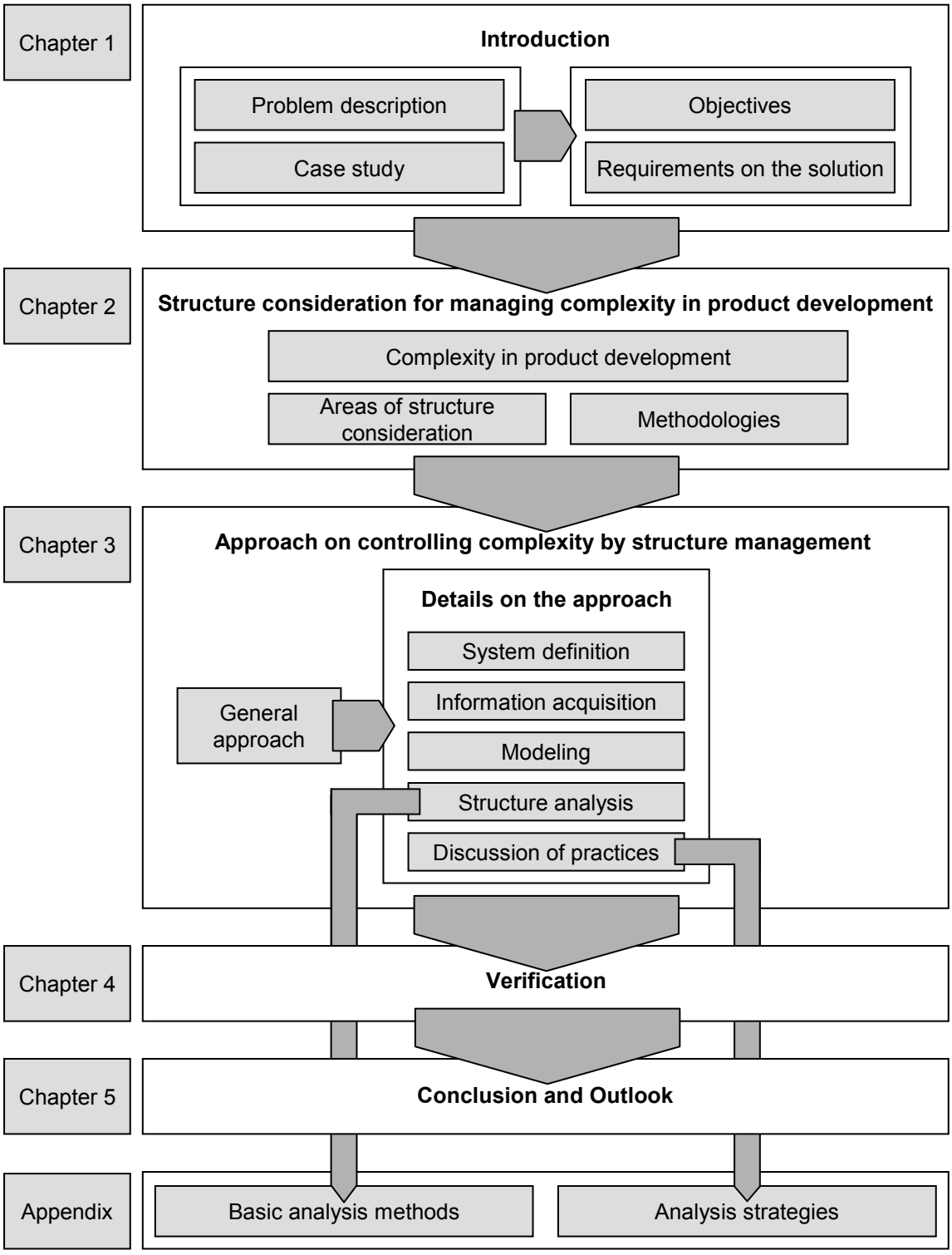


Figure 1-13 Structure of the thesis

2. Structure consideration for managing complexity in product development

Managing complexity is central to many research areas, particularly the consideration of dependency networks, i.e., the structural complexity attracts attention in various scientific works because dependency-based system structures affect system characteristics and behavior [RIEDL 2000, P. 8]. In the following, a short introduction to complexity in product development is presented (Section 2.1). Definitions of related terms and classifications of complexity are described according to their appearance in the literature. Typical problems are presented which result from interacting with complexity followed by a compilation of applied strategies of complexity management. In many approaches complexity is only considered as a negative concomitant of product development; consequently, such approaches aim at avoiding or at least minimizing complexity by suitable strategies. Nevertheless, some researchers mention advantages that can accompany an enhanced complexity, e.g., an increased flexibility in product variety as well as related processes. Thus, the overview points out recent research on the prospects of controlled complexity for future product development.

The thesis focuses on the integration, adaptation, and enhancement of established methods and techniques, and areas of product development are identified at the outset, which focus on dependency structures (Section 2.2). The aspect of engineering change is closely observed due to its relevance to the interaction with dependency networks. Here, the term "engineering change" is defined as any adaptation occurring in product development, i.e., the definition is not limited to the scope of the engineering change management that only represents one facet of product development [LINDEMANN ET AL. 1998]. As in all fields of structure consideration, techniques of information visualization and computational algorithms are required, and a brief overview of the available fundamentals is given.

Existing methodologies for the management of complex engineering data are discussed (Section 2.3) in so far as they are relevant to or precede the scientific approach of the thesis. The multitude of matrix-based approaches is given close consideration, as those approaches are widely applied in several product development methods. These matrix-based approaches are classified in three groups that differ in the number of element types involved. Based on this classification, the systematic enhancement of matrix-based methods and the need for an integral approach to the consideration of multiple element types is shown.

Section 2.4 summarizes the literature and points out the requirements for future structure consideration in product development that has not been sufficiently addressed by published research thus far. In Figure 2-13, the matrix-based approaches are assigned to aspects of complexity discussed in Chapter 1 (see Figure 1-1 and Figure 1-7). The fact that a holistic coverage of dependencies within and between these aspects of complexity is not available provides the starting basis for the approach of this thesis.

2.1 Complexity in product development

Complexity analysis calls for the definition of relevant terms, as these are used differently depending on the specific focus. Therefore, the terms system, structure, and complexity are

defined below, and their meaning within this thesis is specified. Problems and strategies for interacting with complexity are presented and aspects are identified that have not been sufficiently considered so far. Additionally, available material for possible prospects due to an improved control of complexity in product development is compiled.

2.1.1 Definitions and characteristics

System

Originating from the area of system theory[16], a general definition describes a system as a specific way to look at the world [WEINBERG 1975, P. 52]. This has to be further specified in order to be applicable to practical use cases. VESTER describes three different parameters that characterize a system: it must consist of several parts; the parts must be different from each other, and they must be linked to a specific architecture [VESTER & HESLER 1980, P. 27]. This definition means that removing or adding a part to a system results in changes to the system's relations (connectivity), and consequently to the entire system. ULRICH & PROBST define a system as consisting of parts, which are linked to each other; the interaction between these parts influences the system's behavior. ULRICH & PROBST further mention dynamics as an important system characteristic [ULRICH & PROBST 2001, P. 30]. For the authors, complexity, which will be defined below, represents a parameter which characterizes a system.

From an engineering point of view PAHL & BEITZ describe systems as technical artifacts that are artificial, concrete and mostly dynamic and consist of sets of ordered elements, which are interrelated by virtue of their properties [PAHL & BEITZ 1996, P. 20]. In addition, LINDEMANN denotes system borders as well as inputs and outputs that connect the system to its surrounding [LINDEMANN 2007, P. 334]. In the context of systems engineering[17] the "NASA Systems Engineering Handbook" states that system elements interact "in an organized fashion toward a common purpose" [NASA 1995, P. 3]. Besides a system's purpose, this refers to a form of system structure, which is also mentioned by DAENZER & HUBER as an important attribute of each system [DAENZER & HUBER 1999, P. 6].

WASSON presents a comprehensive definition of a system and states that it means "an integrated set of interoperable elements, each with explicitly specified and bounded capabilities, working synergistically to perform value-added processing to enable a user to satisfy mission-oriented operational needs in a prescribed operating environment with a specific outcome and probability of success" [WASSON 2006, P. 18]. Besides the characteristics mentioned, this definition requires further explication. WASSON means by "interoperable elements" that system elements "must be compatible with each other in form,

[16] According to PULM, the system theory relates to the examination, development, and control of systems and their principles both within and between individual sciences [PULM 2004, P. 21]. Here, a system means a circumscribable unit with internal and external dependencies [PULM 2004, P. 19].

[17] Systems engineering and its relevance for this thesis is more closely considered in Chapter 2.3. A short definition of systems engineering states: "The function of systems engineering is to guide the engineering of complex systems" [KOSSIAKOFF & SWEET 2003, P. 3].

fit, and function". It is possible to define system elements as "interoperable and complementary", but the aspect of redundant design for security-relevant systems would be contradictory. With "explicitly specified and bounded capabilities" and "working synergistically", WASSON mentions that the integration and interaction of the elements of a system serve "to accomplish a higher level capability that cannot be achieved as stand alone elements". Furthermore, the purpose of a system is stated in the definition by "satisfying mission-oriented operational needs". Finally, the aspects of uncertainty and risk are implemented as system ingredients by the "probability of success" [WASSON 2006, P. 18].

For this thesis, the following definition is proposed in order to understand systems:

A system is created by compatible and interrelated parts that form a system structure, possess individual properties, and contribute to fulfill the system's purpose. Systems are delimitated by a system border and connected to their surroundings by inputs and outputs. Changes to parts of a system can be characterized by dynamic effects and result in a specific system behavior.

Structure

The term "structure" is directly linked to the above definition of a system. Typically, the structure is seen as an attribute of the system [DAENZER & HUBER 1999, P. 6]. According to MALIK, a structure describes a kind of an order. This means that a multitude of elements are linked in such a way that knowledge about a specific part of the order may generate correct expectations [MALIK 2003, P. 213]. The existence of elements and relations between them automatically results in a structure. For example, the elements of many system structures can be hierarchically classified by their levels of detail. In definitions of systems engineering, the following top-down classification is usually applied: system, segment, element, subsystem, assembly, subassembly, and part [NASA 1995, P. 3]. On each of these levels, structures exist and the intended application determines the requirements.

In product development, a specific focus is product architecture, which describes the dependency structure within components of a product. ULRICH & EPPINGER provide the following definition: "The architecture of a product is the scheme by which the functional elements of the product are arranged into physical chunks and by which the chunks interact" [ULRICH & EPPINGER 1995, P. 132]. Further, they refer to modularity as a characteristic of major importance for product architecture. In this thesis, the term "structure" will be applied instead of "architecture", because the term "product architecture" often refers to hierarchical element orders. As a structure can be any system comprising interrelated elements, a hierarchical product architecture or tree-like product decomposition represents an explicit specification of a structure.

In product development, system structures represent the objective of many analysis methods that reach beyond the characteristics of the modularity mentioned before. Such methods are described by LINDEMANN or NASA ([LINDEMANN 2007, P. 117FF], [NASA 1995, P. 81FF]). Besides hierarchically-ordered structures, the composition of peer linkages can also be characterized by the formation of specific subsets ([WASSON 2006, P. 69], [LINDEMANN ET AL. 2005, P. 110FF]). For example, FOULDS and GROSS & YELLEN describe the discipline of graph

theory, which provides mathematical fundamentals suitable for structural analysis ([FOULDS 1992], [GROSS & YELLEN 2006]).

In this thesis, the term "structure" is understood as the network formed by dependencies between system elements and represents a basic attribute of each system. Structures can be characterized by the specific compilation of implied linkages between system elements and can be divided into subsets. Structures and their subsets can be analyzed by means of computational approaches, primarily provided by the graph theory.

Complexity

Many definitions can be found for the term "complexity", and a number of disciplines focus on its different aspects. To understand "complexity" in terms of engineering, the basic definition from fundamental cybernetics[18] is helpful, as it differentiates between simple, complicated, and complex problems [GOMEZ & PROBST 1997, P. 14FF according to WIENER 1948]. In cybernetics, simple problems are characterized by a few parameters possessing low cross-linking, whereas complicated problems possess a multitude of parameters with intense connectivity. The difference between complicated and complex problems is given by a system's dynamic. The structure of a complicated problem remains stable during a particular time period, whereas complex problems are characterized by a high dynamic of changes.

MALIK names variety, defined as the quantity of different states a system can assume [MALIK 2003, P. 186], as an important characteristic of complexity. MALIK also mentions that complexity originates from interactions between elements and notes the importance of combinatorics for the determination of system complexity.

Even within the engineering discipline, no standardized definition of complexity exists; rather a multitude of specific viewpoints can be identified [PILLER & WARINGER 1999, P. 5FF]. EHRLENSPIEL differentiates between the complexity of technical systems [EHRLENSPIEL 1995, P. 30] and the complexity of objects [EHRLENSPIEL 1995, P. 49F]. The complexity of technical systems depends on the quantity of different elements and their connectivity, i.e., complexity means a measurable system characteristic. In contrast, the term "complicated system" describes the subjective difficulty in interaction with technical systems that often depends on one's personal abilities. EHRLENSPIEL mentions that identical situations can be complicated for one person, but not for another. The complexity of objects is characterized by the following parameters: quantity of variables, connectivity, intransparency and momentum. According to EHRLENSPIEL, most approaches to problem solving aspire to reduce the complexity of objects [EHRLENSPIEL 1995, P. 49F].

A specific view on the complexity of product and system variants is available from FRANKE ET AL., who differentiate between the characteristics already mentioned, connectivity and variety [FRANKE ET AL. 2002]. Here, connectivity means the diversity of relations, whereas variety means the diversity of elements. Connectivity is composed of the types and number of relations, and variety is composed of the types and number of elements. The aspect of

[18] According to ASHBY, cybernetics considers integrated concepts of problem solving and is the "art of steermanship" which requires "co-ordination, regulation, and control" [ASHBY 1956, P. 1].

interdependence of system elements is also mentioned by KÖSTER, who defines complexity as the possibility of comprehension and control of a problem, structure, or system, which depends on the relations of internal and external elements [KÖSTER 1998, P. 21]. The external elements can be combined in the term "environmental complexity", which can be differentiated from the internal complexity of a system [PUHL 1999, P. 1]. From the viewpoint of business administration, the external elements can represent, for example, the origin for the active creation of internal complexity in companies. A company can react to a raise in the external market complexity (e.g., due to a larger diversification of customer requirements) by an increase of the internal complexity of its product program (e.g., by enlarging the variety of offered variants). Besides the aspects of dynamics and the degree of interdependency, PUHL further refers to tolerance and multi-dimensionality as characteristics of complexity [PUHL 1999, P. 4]. Tolerance specifies the sensitivity of a system to the impact of its surrounding. Multi-dimensionality refers to the extent of integrated domains in a system (e.g., functions, components, or processes).

SCHUH & SCHWENK also differentiate origins of complexity by internal and external reasons [SCHUH & SCHWENK 2001, P. 10FF]. They identify the increase of product variants (as internal reasons) as the origin of product complexity that may lead to a detrimental circle of increasing complexity: the increase of product variants (due to covering new niche markets) raises the internal costs of companies (complexity costs) that are followed by an increase of product prices and the decrease of competitiveness; if companies react to this by increasing product variants (in order to reach even more niche markets) the circle is closed. SCHUH & SCHWENK further mention that the increase of product variants is often associated with the decrease of product life cycle time; this effect also results from enhanced market orientation [SCHUH & SCHWENK 2001, P. 14F]. As the reasons for increased complexity emerge from the external side, SCHUH & SCHWENK identify mainly the market, but also norms, standards, laws, and competitors. When considering again the four aspects of complexity mentioned in Chapter 1 (see Figure 1-1), external complexity arises from the market, whereas product and organizational complexity comprise part of the internal complexity. This sequence documents again the importance of the market as the source of complexity (see Section 1.1).

Complexity from the viewpoint of project management is discussed in DANILOVIC & SANDKULL [DANILOVIC & SANDKULL 2002]. In their research, technology, people, and functionality are identified as sources of complexity. In the context of the four aspects of complexity presented in Chapter 1, technology can be assigned to product complexity and people are implied in organizational complexity. As the functionality of an artifact is closely related to its structure (see the above definition of a structure), functionality as a source of complexity is covered by the connectivity of components within the aspect of product complexity. Whereas DANILOVIC & SANDKULL view the understanding of their identified sources of complexity as an analytical issue, they also offer related sources of uncertainty that represent a management issue. Uncertainty can also be seen as a consequence resulting from existing complexity instead of a source of complexity. Consequently, decreasing complexity means less uncertainty in product development. In this context, MALIK mentions structural uncertainty as a characteristic of complexity. This means that not only specific values of variables are unknown, it can not even be determined which variables are relevant and how they are linked to each other in a complex system [MALIK 2003, P. 292].

A comprehensive classification of the types of complexity is proposed by WEBER, who integrates the different aspects mentioned before (see Figure 2-1). He splits complexity into five dimensions and links them to the "components of the technical strategy of companies developing and producing technical products or systems" [WEBER 2005].

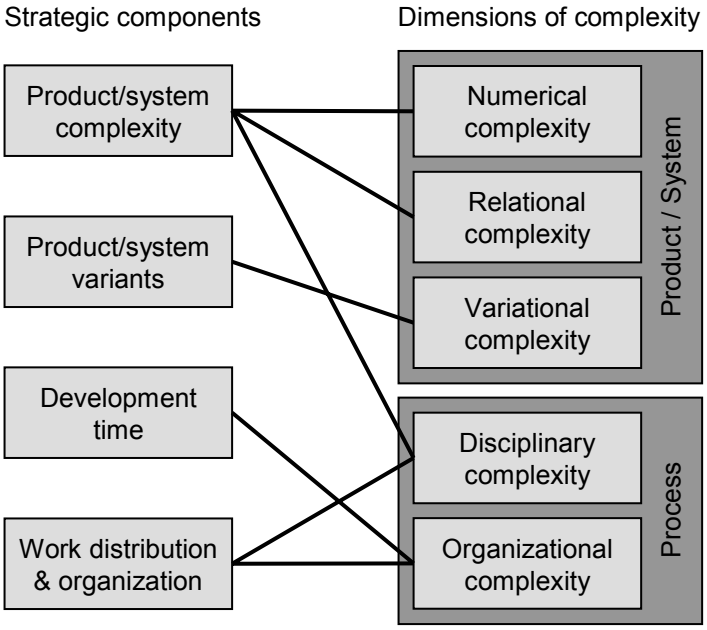


Figure 2-1 Technical strategy and dimensions of complexity (according to [WEBER 2005])

Numerical complexity relates to the number of components in a product or a system, whereas relational complexity describes the number of dependencies between components. Variational complexity describes the number of variants of a product or system and disciplinary and organizational complexity refer to the number of disciplines involved and the distribution of work, e.g., global cooperation. WEBER introduced variational complexity because numerical and relational complexity do not describe possibilities of variant specifications entirely: on the one hand, variants can be "realized individually and independently, i.e., causing numerical complexity but minimal relational complexity"; on the other hand "variants could be realized by combining a minimal set of pre-defined modules, i.e., generating minimal numerical complexity at the cost of increasing relational complexity". Although, WEBER retains the traditional aspects of complexity criteria (numerical and relational), he explicitly refers to a more holistic definition for application to product development by considering further sources (disciplinary and organizational complexity) [WEBER 2005].

The dimensions in Figure 2-1 point at the possible origins of complexity in product development. As already explained, the quantity of elements and relations is one major characteristic for a product's or a system's complexity; however, LINDEMANN mentions that even the product development of simply composed products, e.g., a plastic bag, can be complex, which is implied in the processes of design, production, or distribution [LINDEMANN 2007, P. 8]. This fact is addressed in Figure 2-1 by the process-related dimensions of disciplinary and organizational complexity. Thus, it can be stated that both these dimensions

comprise numerical and relational complexity as well. An interesting aspect arises from linking development time to organizational complexity. That means that the bisection of a product's development time can result in increased organizational complexity, if, for example, concurrent engineering processes must be enforced.

The mentioned characteristics of complexity can be assigned to the interrelated four aspects of complexity, which have been introduced in Chapter 1 (see Figure 1-1). In Figure 2-2 it can be seen that the numerical and the relational complexity characterize all four aspects.

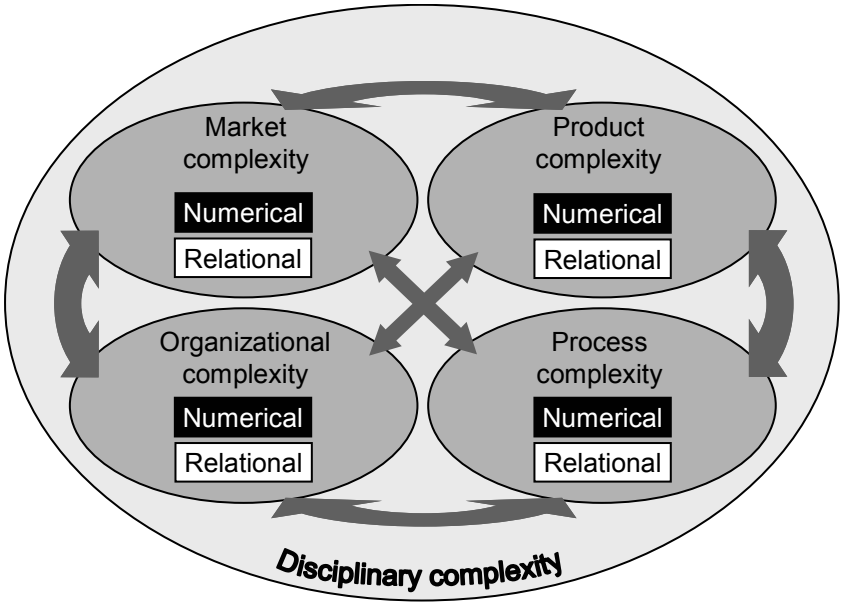


Figure 2-2 Classification of complexity types within the four aspects in product development

In terms of the products, the combination of both forms comprises the variational complexity, as explained by WEBER [WEBER 2005]. In the context of the present thesis, the organizational complexity forms an aspect that also implies numerical and relational complexity. The aforementioned disciplinary complexity is based in the dependency within and between the four aspects. For example, the large number of disciplines involved in the development of mechatronical products results in highly interlinked processes that are executed by several different but interacting organizational teams. The dynamics of complex systems results from cause-and-effect chains formed by the present dependencies within and between the four aspects of complexity.

2.1.2 Problems with handling complexity

As complexity represents a major challenge in all areas of product development, the problems and consequences due to interactions with complexity must be particularly considered. In this context, DÖRNER mainly refers to aspects of dynamic system behavior; he discovered in empirical studies that six general failures can occur when interacting with complex problems [DÖRNER 1992, P. 58FF]. These are incomplete identification of objectives, restriction to

sections, one-side concentration, unconsidered side effects, a tendency towards oversteering, and a tendency towards authoritarian behavior. Due to these problems, users have difficulties interacting intuitively with complex systems and obtaining desired results.

PUHL notices that complex systems can not be analytically determined [PUHL 1999, P. 10]. That means that the application of computational methods does not result in the provision of the optimal action, rather decisions have to be made depending on a specific situation without proof of having achieved the optimum. PUHL even mentions that the complete acquisition of all effect chains is impossible in complex systems [PUHL 1999, P. 10].

Besides problems due to interactions with complexity, some authors focus on problematic consequences that arise from non-manageable complexity in product development. In the context of the complexity of product variety, SCHUH & SCHWENK differentiate between direct and indirect consequences of non-controlled complexity [SCHUH & SCHWENK 2001, P. 19]; they note increasing costs as direct effects, whereas indirect effects can be decrease of enterprise's competitiveness. ANDERSON mentions that the request for product customization becomes continually more important in modern product development [ANDERSON 2006, P. 58FF]. LINDEMANN ET AL. focus on engineering changes and indicate that increasing complexity in product development results in severe problems when enterprises must react to customer-induced adaptations [LINDEMANN ET AL. 2005, P. 107FF]. The authors declare the need to support the analysis and control of product structures by appropriate methods and tools.

In regards to interactions with complex systems, it can be summarized that complexity in product development does not represent a problem per se; rather it is the lack of ability of users to control complexity that leads to severe consequences. Consequently, improved control of complexity allows for enhanced possibilities in product development.

2.1.3 Complexity management strategies

Some of the primary origins for increasing complexity in product development are not influenced by the producing enterprises, e.g., high market pressure due to globalization effects [PINE 1993, P. 33FF]. Whereas several strategies are documented in the literature, the avoidance of complexity can not represent the only strategy for facing problems of complexity. Several authors refer to the term "complexity management", which is often applied in the sense of variant management. Such approaches focus explicitly on correct dimensioning and on controlling the variety of a company's business activities [SCHUH & SCHWENK 2001, P. 34]. This specific focus does not consider the implied complexity of products or processes, i.e., a structure's contribution to fulfill the requirements of a product's functionality; for this reason, strategies and methods of variant management are only partly applicable for developing the thesis approach. Nevertheless, three major strategies for managing complexity can be identified and will be more thoroughly discussed in the following section due to their relevance to structural complexity. Here, the acquisition and evaluation of complex systems represent the basic strategy required for successive avoidance and reduction, as well as for the management and control of complexity.

Acquisition and evaluation of complex systems

Acquisition of complex systems models addresses some of the problems of interaction mentioned above: people often incompletely identify the objectives, restrict themselves to specific extracts of the system, or do not consider existing side effects [DÖRNER 1992, P. 61]. Therefore, the acquisition represents a fundamental step in successful complexity management that is required for all management strategies that focus on downstream phases (evaluation, avoidance, reduction, and control).

For modeling complex processes, mathematically-based methods of data acquisition are described [HANGOS & CAMERON 2001, P. 286FF]. However, only data that can be clearly measured is covered by these approaches. General modeling languages that serve as writing models in a structured language require possibilities of an explicit description as well [HANGOS & CAMERON 2001, P. 484FF]. Complex systems based on experience and knowledge and which contain relations that can not be specified by standardized descriptions require different possibilities of acquisition.

VESTER suggests a criteria matrix approach for acquisition purposes [VESTER 2000, P. 162FF]. This is similar to approaches from several other authors, e.g. STEWARD or PIMMLER & EPPINGER ([STEWARD 1981], [PIMMLER & EPPINGER 1994]). The procedure of acquisition follows a systematic scheme and supports the consequent consideration of all dependencies, as the matrix cells represent all possible direct linkages between implied system elements. For the acquisition process of large-scale impact parameters by means of software support, SABBAGHIAN ET AL. describe a related procedure [SABBAGHIAN ET AL. 1998].

DANILOVIC & BROWNING observed that the chosen method of data acquisition exerts a major influence on the quality of the resulting system structures [DANILOVIC & BROWNING 2004]. They provide an overview of acquired impact matrices[19] that are documented in recent literature. An interesting fact is that for most of the observed structures the process of information acquisition is not documented. The largest matrix contains 60 elements, which does not even correspond to the upper end of data amounts often referred to as complex systems (sometimes comprising several hundred elements or more). BROWNING notes that "there is no absolute limit" to acquisition matrices; however, "practical use provides restrictions" [BROWNING 2001]. An acquisition procedure that implied 140 elements in matrix form is documented by ALEXANDER, who states that it took him months to complete the acquisition [ALEXANDER 1964]. Besides the extensive efforts required for the compilation of large matrices, JARRATT discovered that the autonomous compilation of impact parameters by designers can lead to unsatisfying results [JARRATT 2004, P. 121FF]. In his research, he executed an exercise with engineers, who had to individually acquire the parts of a diesel engine and depict them in a matrix. Results showed highly varying differences in the system layout, especially as different linkage types were considered.

Concerning the evaluation of complex structures RAPP defines variant-related and structure-related complexity indices that permit characterizing structures [RAPP 1999, P. 42]. Besides

[19] These impact matrices correspond to the Design Structure Matrices (DSMs) that are further discussed in Chapter 2.3.2.

the amount of the implied components, four indexes directly refer to the quantity of product variants and their compilation. These indexes are meant to apply to variant management and do not serve to characterize general system structures. SCHUH & SCHWENK do not restrict themselves to index values but describe four fundamental types of structures that can appear as subsets of complex structural constellations. These are series, modules, building blocks and packages [SCHUH & SCHWENK 2001, S. 81]. The authors describe the advantages and disadvantages of these subsets for a product design that suits the requirements for variant management. However, they note that a distinct classification of a product's structure to one of these types is often impossible, but a product's structure is typically compiled by a mixture of these generic subsets. Based on the axiomatic design approach [SUH 1988] EL-HAIK & YANG suggest information measures and derived mathematical relationships to quantify single components in a structure [EL-HAIK & YANG 1999]. In their findings, the authors assume that the measurement of components in intensely linked systems requires possibilities of characterization that are so far not available but are necessary for practical application. An interesting approach to the evaluation of complex structures is a design landscape containing physical components and a choice of architecture interconnections [BRABAZON & MATTHEWS 2002]. The approach provides a theoretic framework for investigating the impact of different modular structures on the process of product design. Here as well, practical use requires the enhancement of specifically applied methods and analysis support.

VESTER complements the criteria matrix (meant for acquisition purposes) by an impact matrix for the evaluation of structures [VESTER 2000, P. 196FF]. The applied methods correspond to approaches from systems engineering, mainly the Design Structure Matrix (DSM), which has been introduced by STEWARD as a method for analyzing the structure of the system design process [STEWARD 1981]. A set of structural analysis criteria can be identified by these matrix approaches, and comprehensive possibilities of interpretation are available. For example, SOSA ET AL. describe the identification and the impact of modular and integrative design on team organizations [SOSA ET AL. 2003]. STRONG provides metrics for evaluating modular design and SHARMAN & YASSINE additionally describe the identification and interpretation of the structural criteria bus and hierarchy ([STRONG 2002, P. 45FF], [SHARMAN & YASSINE 2004], [BALDWIN & CLARK 2000]).

The application of a square matrix for system evaluation is popular and permits a comprehensive analysis. Therefore, DSM approaches are considered in detail in Section 2.3. Besides a matrix analysis, specific approaches for evaluating variant complexity apply tree structures typically to represent the emerging variants that occur due to subdividing attribute specifications ([LINDEMANN ET AL. 2006, P. 231FF], [SCHUH & SCHWENK 2001, P. 100FF]). Using an automotive case study LINDEMANN ET AL. further demonstrate the application of an ABC-analysis as a useful method for characterizing elements in complex systems [LINDEMANN ET AL. 2006, P. 224].

It can be noted that several methods for the acquisition of system linkages are founded on matrix-based approaches. If acquired data can not be formulated in mathematical descriptions, matrices are often applied for capturing qualitative dependencies. Large amounts of data make it difficult for users to keep the system overview, and the resulting quality of acquired models largely depends on available resources and the individual viewpoint of the

interrogated users. As the acquisition of system structures represent the basis for all further steps in structural investigations, an appropriate methodology is required that assures an efficient capturing process and high quality results. For evaluating the acquired complex structures, index values are provided that relate to fundamental structural characteristics and focus on variant trees. Further approaches consider basic structural subsets for characterizing complex systems but do not allow for interpretations when the subsets appear in combination. Although some analytical methods are on hand, practical applications of structure evaluation ask for more powerful analysis approaches and possibilities of case-specific interpretation.

Avoidance and reduction of complexity

An effective strategy for the management of complexity is avoiding it. In general, the complexity of each system can be reduced, if it is possible to eliminate elements and relations while keeping the existing system functionality. However, such actions mean adaptations to the system, and it must be guaranteed that changes can be applied in practice, if they seem to be useful from a structural point of view. In any case, complexity often exists, that is unnecessary to fulfill the system's purpose and the customer's requirements. Consequently, this complexity can be avoided. Several authors describe related approaches focusing on variant management. E.g., the definition of variants as late as possible in the value creation chain is mentioned by FIRCHAU and RAPP ([FIRCHAU 2003, P. 9], [RAPP 1999, P. 49]). The later the variants are defined in the development process, the less the complexity that has to be handled. RAPP also describes the elimination of rarely sold products from the product program [RAPP 1999, P. 65]. However, these strategies are not useful if complexity arises from the system itself, i.e., its structural design.

A possibility of reducing structural complexity is available by the tearing[20] approach ([DSM INTERNET PAGE 2007], [YASSINE 2004]). Here, dependencies between system elements are identified, whose avoidance permits the optimization of the modular design and the reduction of system complexity overall (by shortening feedback loops). However, the mathematical base of the tearing approach is weak and achievable results depend on several constraints. Tearing is further considered in Section 3.6 of this thesis, and Section 4.1 provides an example that illustrates the limits of applicability of this approach.

If complexity can not be avoided, often an optimized product structure can help reduce unnecessary product complexity. Most related approaches aim at a modular product design, described by ERICSSON & ERIXON and BALDWIN & CLARK ([ERICSSON & ERIXON 1999], [BALDWIN & CLARK 2000, P. 149FF]). The objective is to design subsystems that are generally independent from each other and only possess a high degree of internal linkages. Practical applications of complexity reduction are the creation of building blocks, platforms, or

[20] Tearing tries to eliminate the dependencies of a structure in order to minimize the length of feedback loops in a system. Additionally, tearing can be used for optimizing the modular design of a system. The approach is based on a partitioning algorithm, which accumulates dependencies in matrix representation at one side of the diagonal or at least as close as possible to the diagonal. Dependencies that can not be aligned close to the diagonal depict objectives for elimination by the tearing approach. Tearing is further considered in Chapter 3.6.

standardized interfaces. The benefit from these measures is the easier adaptation of system elements, as the resulting impact to the system is mostly limited to the created modules.

Management and control of complexity

In contrast to the objective of avoiding complexity, PUHL also mentions its increase as an important strategy of successful complexity management. He describes the existence of an optimal complexity value of systems [PUHL 1999, S. 20FF]. This is generally determined by the environmental complexity of a system, e.g., the market requirements. A specific level of complexity can be useful to permit the flexibility of process structures that can be specified to adaptations in the system's surrounding, and therefore provide competitive advantages ([PUHL 1999, P. 23], [MAURER & LINDEMANN 2007]). If the right level of complexity is determined, the existing complexity may be controlled by the use of adequate methods.

PUHL states that controlling complexity stands for the ability to handle the complexity of processes and their effects without jeopardizing the targets [PUHL 1999, P. 23]. He further classifies the effects into unexpected disturbances or failures, as well as complex structures or interfaces. The related definition of complexity management is applied by PUHL as the adequate handling of complexity in order to realize the targets and to make use of the potential of complexity compositions. Here, PUHL focuses only on process complexity and does not cover structural complexity in a product's design [PUHL 1999, P. 23].

In contrast to PUHL, SCHUH & SCHWENK consider products and resources of complexity management for large quantities of product variants [SCHUH & SCHWENK 2001, P. 34]. They define controlling diversity in all sections of value creation as the main target of complexity management, as this can maximize the benefit to the customer by simultaneously increasing efficiency for the producer. SCHUH & SCHWENK propose different strategies of complexity management for the entire enterprise [SCHUH & SCHWENK 2001, P. 52FF]. Whereas these strategies (trade off between economy of scale and economy of scope, customization versus standardization, and mass customization) can not be transferred directly to specific tasks in product development, GOMEZ & PROBST provide a five-step model of an integral problem-solving method applicable to connected system environments [GOMEZ & PROBST 1997, P. 27FF]. As it can be seen in Figure 2-3, the model steps are described on a generic level. However, the implementation of the first two steps explicitly addresses the acquisition of system elements and their dependencies, as well as the identification of structural characteristics [GOMEZ & PROBST 1997, P. 35FF].

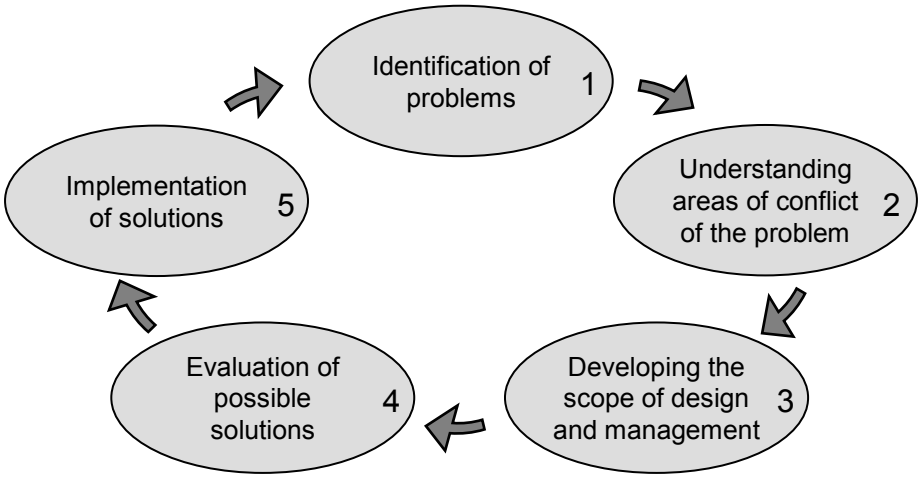


Figure 2-3 Five steps for the integrated problem solving (adapted from [GOMEZ & PROBST 1997, P. 27])

DAENZER & HUBER describe four general principles for the compilation of product systems based on structural characteristics [DAENZER & HUBER 1999, P. 22F]: the minimization of interfaces and the modular system design, which are widely applied in system engineering; the "piecemeal engineering" means to execute changes in complex systems by small steps that can be undone if necessary. The principle of minimizing decision anticipation aims at solutions that possess the least constraint for the entire system. These principles can be applied in the development step of the problem-solving approach presented.

Little research considers strategies of complexity management from a structural point of view. Generic approaches for controlling process dependencies can only be partially applied to problems of complexity in product development.

2.1.4 Opportunities of controlled complexity

Some of the preceding remarks can lead to the assumption that complexity is harmful per se. As mentioned before, only a few authors have considered the necessity of choosing the right level of system complexity, e.g., assuring adequate flexibility of customer-induced adaptations [PUHL 1999, P. 20FF]. MAURER & LINDEMANN note that "it may not be advantageous to reduce complexity at any cost. Complexity often relates directly to attributes relevant to the customer; thus complexity reduction may decrease competitiveness" [MAURER & LINDEMANN 2007]. PINE ultimately defined the term "mass customization" and described the potential of increased product variation, i.e., increased complexity, in several industries [PINE 1993, P. 33FF]. The core idea of mass customization is an optimal combination of mass production with customized product specifications, i.e., turning low internal complexity into high external complexity for specific customer requests. The better enterprises can control their existent complexity the more customization becomes possible to market quickly and at reasonable costs.

Competitive chances of complexity also arise from the fact that controlled product complexity can serve as a barrier to product plagiarism [WILDEMANN ET AL. 2007, P. 50]. The main idea of such an approach is to develop a comprehensive product spectrum instead of discrete products (see also Section 1.1). Individual products can be derived from such a product spectrum by specifying attributes. MAURER & LINDEMANN describe the approach as follows: "The benefits of a controlled structural complexity are invisible to potential product imitators. They permit fast and efficient product adaptations without displaying this knowledge in the product itself. For this reason, reverse-engineering of available products will not uncover the internal basis enabling a company to specify new product variants. Competitors can imitate the specifications of the product delivered to the market with a delay caused by the reverse engineering process. However, they can not copy the core competence: being able to produce customized products as required by the market within a short space of time. Hence, the ability to control complexity rather than reducing it can be seen as a major competitive advantage" [MAURER & LINDEMANN 2007].

CLARKSON ET AL. mention that an effective system for controlling complexity can predict the impact of change, which previously would have been unnoticed [CLARKSON ET AL. 2001]. As newly arising requirements are generally realized by adapting existing products, change prediction helps shorten development cycles and results in less development time [CROSS 2000, P. 9]. Hence, the possibility of controlling change dependencies in product development may allow more comprehensive adaptations, as the resulting consequences can be quickly identified or are even known immediately.

RIEDL as well as LINDEMANN ET AL. mention that a multitude of system features occur because of the system structure and are not applied to discrete components ([RIEDL 2000, P. 8], [LINDEMANN ET AL. 2005]). For example, system robustness to adaptations represents a feature emerging from the assembly of system elements and their dependencies. Furthermore, the demand for systems that perform robustly when system changes appear is directly related to the control of their complex dependency networks. The better system complexity can be controlled, the better the structure can be adjusted in order serve the desired functional objectives.

2.2 Areas of structure consideration

In the following an overview of the objectives of established structure-based investigations in product development is given from different perspectives. Additionally, related fundamentals on information visualization and computational procedures are mentioned, which are relevant for the thesis approach presented later.

2.2.1 Objectives

BROWNING mentions a better understanding of complex systems as a primary target of structure considerations [BROWNING 2001]. This is based on the procedure of dividing a system into subsets, capturing their mutual relationships, and capturing the system's inputs and outputs. Once developers divide an accessible system structure, the consequences of the specific system impact and system behavior can be estimated [LINDEMANN ET AL. 2005]. This

means that the knowledge about a system's structural compilation allows for a better prediction of its behavior. For example, SHARMAN & YASSINE show that structural product attributes contribute to a large extent to system behavior [SHARMAN & YASSINE 2004]. Besides these authors, RAPP and BALDWIN & CLARK describe structural product attributes for characterizing system behavior ([RAPP 1999, P. 18FF], [BALDWIN & CLARK 2000, P. 63FF]).

Every interaction in the development process can be seen as an engineering change executed in a system. LINDEMANN ET AL. define an engineering change as an action introduced because results "do not meet the existing demands of the product" [LINDEMANN ET AL. 1998]. The authors remark that "impulses for changes occur unrelated to time and content" and that ineffective changes result in "money and time-consuming alterations". CLARKSON ET AL. mention the identification of possibilities and risks of planned adaptations as a practical application for managing engineering changes [CLARKSON ET AL. 2001]. If an engineering change is implemented for a product, the determination of required subsequent adaptations due to this initial adaptation must be supported [CLARKSON ET AL. 2000]. FLANAGAN ET AL. describe a model of change propagation, where change propagation means impacts to the system elements of a product design due to one specific adaptation which causes the need for successive product changes [FLANAGAN ET AL. 2003]. The authors state that an analysis method is needed, which illustrates change propagation paths in order to allow for change management during product redesign. Further product-related use cases of engineering changes are, for example, modifications to the compilation of components due to the variant design. CROSS states that the majority of design activities relate to variant design, as "in most cases the designer is asked to design something similar to that which he or she has designed before" [CROSS 2000, P. 9]. DEUBZER ET AL. provide a comprehensive view of the engineering changes for technical products in German industries [DEUBZER ET AL. 2005], where the focus is on subsequent changes to work results already released in product development.

A specific application of path finding in complex product development structures is represented by the signposting method [CLARKSON ET AL. 2000]. Here the next most appropriate design task is suggested in order to improve the design process. Suggestions are based on structural information, e.g. mutual task dependency or independency, and provide potential for individual adaptation. With the focus on process structures, several authors also mention the potential of increased process efficiency by re-designing their structural compilation, mainly by reducing implied process iterations ([GROSE 1994], [KRISHNAN ET AL. 1997]).

Another application field of engineering changes is provided by the trend towards product customization, such as the strategy of mass customization. PINE describes the focus, goals, and key features of mass customization compared to conventional mass production for several use cases like the automobile or banking industry [PINE 1993, P. 31FF]. The increase in the number of product variants consequently results in more adaptation effort. PINE presents strategies to keep engineering changes at a minimum [PINE 1993, P. 196FF], e.g., bus-modularity or cut-to-fit-modularity. These strategies relate directly to the product structure that must be adapted in order to fulfill the strategic goals. PILLER mentions the importance of structural approaches for complexity avoidance and reduction and to control a successful

strategy of mass customization with a focus on product specification [PILLER 2001, P. 225F]; and he provides a set of appropriate measures on a generic level.

The consideration of system structures help to assure successful product development, because active structure deployment instead of passive structure formation creates a product according to its specific intention. When focusing on product components, the active creation of modules is often applied, whereas modularization strategies are based on the structural optimization of the component dependencies [BALDWIN & CLARK 2000, P. 63FF]. For this reason, the dependencies between subsets of system elements have to be minimized by integrating dependencies with internal modules. As a positive result of product modularization, BALDWIN & CLARK mention the improved development of complex products, as fewer dependencies have to be considered [BALDWIN & CLARK 2000, P. 5F]. LINDEMANN ET AL. mention that adaptation time and costs can be reduced and system reliability can be increased if adequate system structures are actively designed [LINDEMANN ET AL. 2005].

The objectives of structure consideration, especially methods for the determination of relevant domains and for a systematic information acquisition, have not been sufficiently covered so far. However, particularly in multi-disciplinary product development it is important to know about the aspects that have to be considered. Furthermore, no systematic approaches for the condensing and selection of individual system views are on hand. When interacting with systems comprising multiple domains, e.g., interrelations of components, processes, and people, having systematic approaches poses an important requirement to control complexity. Several analysis methods for complex system structures are documented in literature but their applicability to comprehensive system structures needs to be enhanced. This would allow for the improvement of structural optimization and support the proactive structure design in product development.

2.2.2 Fundamentals

Structure considerations in product development are based on certain fundamentals. These are mainly concerned with the form of data representation and applied computational techniques. In the following a short introduction to basic methods and procedures is given, which are relevant for the thesis approach and provide a sound basis for the discussion on methodologies.

Information visualization

Several research disciplines (e.g., systems engineering, operations research, data mining, or system dynamics) concern questions of visualizing complex system data. The general objective of these representations is to permit users a global system overview as well as provide focused views on specific aspects in order to obtain a better system understanding.

WARE mentions general gestalt laws that describe “the way we see patterns in visual displays” [WARE 2004, P. 189FF]. According to this, the fundamental elements’ entities and relationships as well as their possible attributes must be represented [WARE 2004, P. 23FF]. For this task, typically diagram or matrix representations are applied. WARE provides an

overview of the basic grammar of node-link diagrams, generally illustrated as digraphs[21] when representing structural data [WARE 2004, P. 210FF]. Applications for visualizing, for example, the general dependencies in economic systems are presented by PROBST & GOMEZ for a variety of industrial use cases [PROBST & GOMEZ 1991]. Graphs can offer the same information as matrices, and both forms can be translated from one to the other [ANDRÁSFAI 1991, P. 133FF]. SHARMAN & YASSINE compared matrix and diagram approaches for describing product architectures and mentioned their specific advantages in practical use cases [SHARMAN & YASSINE 2004]. The authors also prepared an overview of visual models applied in different areas of engineering. Despite the possibility of mutually switching between matrix and graph representations, information losses can still occur [BROY 1998]. Such losses arise from additional information going beyond the pure connectivity of elements. If, for example, the alignment of elements in a matrix visualizes a cluster, this alignment can not be easily transferred to a graph representation. Therefore, information about the system structure should be stored separately and only be changed into graphs or matrices for representation purposes.

Regarding matrix representations, MALMQVIST provides a classification of related methods characterized by their scope as well as their content [MALMQVIST 2002]. This classification covers element and product-level matrices and takes the different types of modeled elements and relations into account. Matrix-based methods are often applied in product development and are mentioned in the literature under a variety of different names. For example, KUSIAK refers to incidence matrices, and ULRICH & EPPINGER refer to interaction matrices; both relate to linking system elements through relationships ([KUSIAK 1999, P. 33FF], [ULRICH & EPPINGER 1995, P. 144]). KUSIAK further provides comprehensive possibilities for optimizing the matrix alignment, when adequate element orders are the basis for mediating structural content for users [KUSIAK 1999, P. 33FF]. In Quality Function Deployment (QFD) or Axiomatic Design, for example, matrices are applied which combine two different element types by dependencies, described by AKAO and SUH ([AKAO 1992], [SUH 1988]).

A multitude of visualization approaches use digraphs, which can be classified by the method of their element positioning. General interaction graphs do not determine the specific location of graph elements, and users are free to arbitrarily arrange those [ULRICH & EPPINGER 1995, P. 145]. Such graphs are easy to create and are often applied to compile initial system dependencies [COYLE 1996, P. 18FF]; however, they are not adequate for larger data amounts, as comprehensibility gets more difficult without any ordering criteria [LINDEMANN 2007, P. 72].

Mind maps represent specific graphs and are used for initial structuring purposes, but also provide rules for the alignment of elements [BUZAN & BUZAN 2002, P. 123FF]. Basically, mind maps are hierarchically arranged; starting from an initial root element further branches can be created. Modeling such tree structures is suitable, for example, for representing the decomposition of hardware or software products ([PAHL & BEITZ 1996, P. 445], [BRUEGGE &

[21] A digraph is a structure which consists of a set of vertices and a set of edges; each edge is incident to the elements of an ordered pair of vertices. The first endpoint of an edge represents the start-vertex of the edge and the second represents the end-vertex [EVEN 1979, P. 3].

DUTOIT 2000, P. 166FF]). Mathematical descriptions of tree graphs can be taken from DI BATTISTA ET AL., but typically have not been applied to mind maps so far [DI BATTISTA ET AL. 1999, P. 41FF]. Specific forms of hierarchically-ordered graphs are, for example, sunbursts or hyperbolic trees that arrange the tree elements radially around the centered root element [WARE 2004, P. 341].

A powerful method for the alignment of highly interlinked structures is represented by a strength-based or force-directed graph [DI BATTISTA ET AL. 1999, P. 303FF]. The advantage of this approach is a graph alignment which is intuitive for users, as relevant embedding of elements in the structure is clearly visualized. Graph nodes repel each other and are mutually attracted by graph edges. It is possible to assign weights to graph edges, representing, for example, physical forces or the amount of information flow. Edge weights then result in different attraction forces (leading to different element distances) between nodes.

Methods for visual representation of large amounts of data using maps are widely applied in data mining and also help users better understand system characteristics [FAYYAD ET AL. 2002]. Visualization methods going beyond matrices and graphs generally do not describe direct interdependencies between system elements but rather their relative specification concerning their shared attributes [SOUKUP & DAVIDSON 2002, P. 5F]. Therefore, these representation forms are not helpful for modeling dependencies between system elements.

Generally, matrices and graphs are applied for visualizing dependency information in product development. Whereas the mutual transferability of both forms is mathematically formulated, only a few applications make use of their combination in order to benefit from the advantage of both visualizations. In regards to dependency representations in graphs, the application of strength-based models provides noticeable enhancements to common interaction graphs, but those models have not been implemented for methods in product development so far.

Computational approaches and algorithms

Computational approaches for the consideration of complex structures are applied for different purposes in product development [KUSIAK 2000]. Here, the focus is on techniques that allow for analysis, control, and optimization of connected system elements.

Neural Networks are inspired by biological information processing and are typically used when explicit knowledge about the problem is unavailable. KUSIAK provides a comprehensive overview on Neural Networks from an engineering point of view [KUSIAK 2000, P. 347FF]. Neural Networks can represent complex problem situations and compute solutions by various techniques, even if deterministic algorithms fail. However, they are not suitable for generating a better understanding of dependencies and system characteristics for users and do not allow direct user interaction with the system structure. Neural Networks can generally provide possibilities for solving optimization problems for structural design.

Evolutionary algorithms (EAs) adapt the mechanisms of evolution and genetics to optimize complex systems. Two major models of evolution which have turned out to be very good for computer simulation and IT applications are evolutionary strategies and genetic algorithms ([RECHENBERG 1973], [GOLDBERG 1989]). They were developed for the optimization of technical systems that contain large numbers of parameters and are suitable for finding high

quality results in adequate time. Randomly selected parameters are varied by recombination and mutation. Resulting structures are rated by a fitness function, which represents the major issue when designing an EA. This function has to describe the system characteristic that must be improved. If the applied EA creates variants of the initial system, the rating computed by the fitness function must identify improvements or deterioration of the characteristics of the system considered when compared to the initial structure. Based on this comparison promising variants can be selected. A lot of standardized EAs are available and can be easily adapted to specific requirements, if an appropriate fitness function can be defined. EAs help to analyze and optimize structures, and therefore make part of complex structure interaction, but do not serve for a user's access to and interaction with structures. In the present thesis approach an evolutionary algorithm is applied for improving a system's overall criticality. This algorithm is described in detail in Section 3.7.

The algorithmic graph theory represents an area of mathematic science and focuses on characteristics of graphs composed of nodes and edges. BOLLOBÁS for example provides a sound introduction to the topic [BOLLOBÁS 1990]. FOULDS describes a collection of application scenarios that provide a good understanding of the scope of applicability [FOULDS 1992, P. 193FF]. Many algorithmic procedures are based on algorithmic graph theory, for example, the identification of paths between elements; this identification of paths provides the basis for techniques of project scheduling [GROSS & YELLEN 2006, P. 493FF]. In fact, several matrix-based algorithms applied in engineering (see Section 2.3.2) originally emerge from algorithmic graph theory [MAURER & LINDEMANN 2007]. In Section 2.3.1 the graph theory is considered in the context of methodologies that are applied to the thesis approach presented here. Applications of graph theory criteria for complex engineering structures are discussed in Section 3.5.3 and are listed in the Appendix for easy access.

Statistics also belongs to mathematical science and is often applied to engineering tasks. For example, in data mining, statistics are applied for analysis as well as presentation of data. KUSIAK provides an overview to data mining in engineering and also refers to applications of pattern recognition in large amounts of data [KUSIAK 2000, P. 498FF].

Matrix-based computations are widely applied to complex systems in engineering. BROWNING provides a sound overview of available techniques and applications of the Design Structure Matrix (DSM) [BROWNING 2001]. These will be examined in Section 2.3.2 together with further relevant matrix approaches. The most commonly used computation concerns the clustering of dependencies in a matrix. The mathematical basis for this application is presented by HARTIGAN, and DANILOVIC & BROWNING present practical applications for project management ([HARTIGAN 1975], [DANILOVIC & BROWNING 2004]). KUSIAK describes a clustering technique that is based on the serial realignment of matrix columns and rows and presents a step-by-step procedure of such a technique [KUSIAK 1999, P. 70FF]. Matrix-based clustering in engineering applications aim at optimized element alignments that allow for the visual identification of closely related groups of elements ([STEWARD 1981], [KUSIAK 1999, P. 70FF]). Here, the realignment represents the application of fundamentals from graph theory to matrix-based visualization.

2.3 Methodologies for interacting with complex engineering data

Several research disciplines provide possibilities for the application of complex problems within the scope of product development. At first, a brief overview of these methodologies is given and forms the background of this thesis (see Figure 1-12). Due to the relevance of matrix-based approaches, their application is considered in-depth below.

2.3.1 Overview of methodologies

Design methodology & systems engineering

In a holistic context HUBKA & EDER define design science as "a system of logically related knowledge, which should contain and organize the complete knowledge about and for designing" [HUBKA & EDER 1996, P. 73]. For practical applications, appropriate methods are required; a comprehensive compilation of 81 methods is available from LINDEMANN [LINDEMANN 2007, P. 239FF]. These methods can be applied to different tasks in product development classified in different phases, e.g., by the Munich Procedural Model [PONN & LINDEMANN 2005]. Throughout all these phases, interaction with complex interrelated data is important and is realized by several matrix-based or graph-based methods. Thus, 20 of the methods described by Lindemann directly apply graph-based or matrix-based linking of elements in the form of, for example, impact matrices, consistency matrices, or functional modeling. Several more methods apply, at least indirectly, networks of interconnected elements. Whereas these methods are based on common principles (e.g., connecting two groups of elements by use of a rectangular matrix), the application possibilities are not matched between them. Therefore, a superior approach is required that allows integrating and transferring differently applied techniques. An example clarifies this statement: for the application of consistency matrices, LINDEMANN mentions the possibility of applying cluster analyses in order to identify closely related groups of elements [LINDEMANN 2007, P. 274]. In the OFD method an identically compiled matrix exists as well, and the intention of clustering could be useful as well, but has not been provided so far. A similar example represents the possibility of feedback loop identification, i.e., the determination of dependencies between system elements that form a closed loop. KUSIAK describes the identification of loops for process modeling by use of square matrices [KUSIAK 1999, P. 36FF]. For the method of functional modeling, systematic feedback loop analysis would be helpful as well, but has not been used so far.

The discipline of systems engineering overlaps with the design methodology mentioned before and is an interdisciplinary approach for the successful development and realization of user-defined systems. Due to the holistic approach, systems engineering is applied to complex problems including many different disciplines. Several organizations, such as INCOSE[22] and NASA[23], provide definitions and models of systems engineering approaches ([INCOSE

[22] International Council on Systems Engineering (INCOSE)

[23] National Aeronautics and Space Administration (NASA)

2002], [NASA 1995]); additionally, a good overview is given by BLANCHARD or KOSSIAKOFF & SWEET ([BLANCHARD 2004], [KOSSIAKOFF & SWEET 2003]). In modern product development, a multitude of concepts and methods are applied that can be found in systems engineering as well. For example, concepts of structuring complex systems have been implemented in strategies of variant design [SCHUH & SCHWENK 2001, P. 32FF]. Several target-oriented design strategies apply the active creation of structures in integral blocks or standardized frames and interfaces, for example, the design for assembly [ANDREASEN ET AL. 1983, P. 119FF].

Design methodology and systems engineering integrate methods and techniques from several disciplines and apply them to product development. The need for managing complexity in product development has resulted in a variety of available methods; using such methods for the analysis, control, and optimization of complex systems can help improve their applicability.

Operations research

The scientific field of operations research provides methods for analyzing and interacting with highly linked structures which exceed the approaches of intuitive problem solving. The PERT method is one such example and must be mentioned due to its close relation to DSM techniques [DANILOVIC & SANDKULL 2002]. This approach is also known as the Critical Path Method (CPM) and is intended to optimize the process of activities in projects in terms of time planning and capacity utilization [DANILOVIC & SANDKULL 2002]. The PERT approach determines the critical path and buffer time for avoiding delays. Operations research uses statistics, data mining, and mathematical simulation approaches to optimize areas such as logistics, decision making, or scheduling problems. Therefore, the application of quantified data is mandatory [WINSTON 2004, P. 5F].

Data mining

Data mining applies mathematical and statistical methods to large amounts of data and generally aims at classifying data and detecting significant patterns. An introduction to engineering-related applications can be taken from KUSIAK [KUSIAK 2000, P. 498 FF]. Besides the analytical possibilities, methods of data visualization are of special interest for interacting with complexity in product development. An overview of related techniques and tools for large-scale data amounts are described by SOUKUP & DAVIDSON and FAYYAD ET AL. ([SOUKUP & DAVIDSON 2002], [FAYYAD ET AL. 2002]). Analysis methods from data mining, which represent data in diagrams, have been introduced to product development. For example, the representation of product components in a portfolio, where the axes depict the outgoing (active) and incoming (passive) dependencies of components, can characterize their criticality in the entire product ([LINDEMANN 2007, P. 289], [AMBROSY 1996, P. 52]). Cluster analysis has been introduced to methods of product development by GAUSEMEIER ET AL. [GAUSEMEIER ET AL 1995, P. 273FF]. Like applications of operations research, prior data is needed for data-mining analysis.

System dynamics

Cybernetics can be seen as the precursor of system dynamics and describes approaches that consider the enormous complexity of corporate interrelations and aim at integrated concepts of problem solving. Fundamentals have been provided by ASHBY and WIENER ([ASHBY 1956], [WIENER 1948]). ASHBY defines cybernetics as "the art of steermanship" which requires "co-ordination, regulation and control". He further mentions that "Cybernetics […] treats, not things but ways of behaving" [ASHBY 1956, P. 1]. The generic ideas of cybernetics have been adopted by other research disciplines; for example, FORRESTER applied them to economic problems [FORRESTER 1961]. TSIEN provides early applications of cybernetic approaches to engineering problems [TSIEN 1955].

In 1961 FORRESTER introduced the term "system dynamics" and "defined it as the investigation of the information-feedback characteristics of [managed] systems and the use of models for the design of improved organizational form and guided policy" ([COYLE 1996, P. 9] according to [FORRESTER 1961]). COYLE describes the development of system dynamics in a logical sequence starting with the application of formal analysis to warfare and business management in the 1940s [COYLE 1996, P. 1]. He indicates that the developed methods were "formalized into the disciplines of operational research and management science" and "have proved to be excellent and powerful for dealing with […] problems […] in which the ability of a system to adjust to changing circumstances as time passes is not a significant aspect of the management problem" [COYLE 1996, P. 1]. COYLE clarifies the importance of considering a system's dynamics by using the example of the management policies of a company: "How can its policies be designed so that the firm becomes robust against change […]?" [COYLE 1996, P. 3].

A basic model of a feedback in system dynamics can be seen in Figure 2-4. A specific state of a system results as consequence of an initial choice, e.g., a decision made by the management of a company. The delay which typically occurs between the choice and the resulting state is depicted by the variable D_S in Figure 2-4. Subsequently, the adopted state becomes known to people, such as the controllers (and also requires some delay D_K). Finally, the acquired knowledge leads to further choices (after the delay D_C) to optimize the actual system state. The entire loop describes the basic dynamic behavior of systems over the time.

FORRESTER notes that "symptom, action, and solution are not isolated in a linear cause-to-effect relationship, but exist in a nest of circular and interlocking structures wherein an action can induce not only correction but also fluctuation, counterpressures, and even accentuation of the forces that produced the original symptoms of distress" [FORRESTER 1980, P. 13]. The interlocking structures mentioned are of major relevance, as a "ubiquitous presence of feedback loops" can be found in comprehensive systems [COYLE 1996, P. 6] that are mutually dependent and interactive. In system dynamics, feedback loops are characterized as goal-seeking or negative[24] feedback loops and growth-producing or positive feedback loops

[24] Instead of the term negative feedback loop, the expression balancing feedback loop is often found in the literature. This has to be handled with care as the (mostly desired) balancing behavior of these kinds of feedback loops does not result in either case [FORRESTER 1980, P. 13F].

[COYLE 1996, P. 6ff]. Although the definition suggests the benefits or drawbacks of the loops, FORRESTER states that "the richness of potential behavior of even relatively simple configurations of feedback loops has not been codified" [FORRESTER 1980, P. 13]. He indicates that "positive feedback loops are usually thought of as being destabilizing; […] A positive feedback loop may be so situated that it strongly stabilizes the oscillatory tendencies of an accompanying negative feedback loop. […] By contrast, actions that superficially appear to be stabilizing can actually be destabilizing" [FORRESTER 1980, P. 13F].

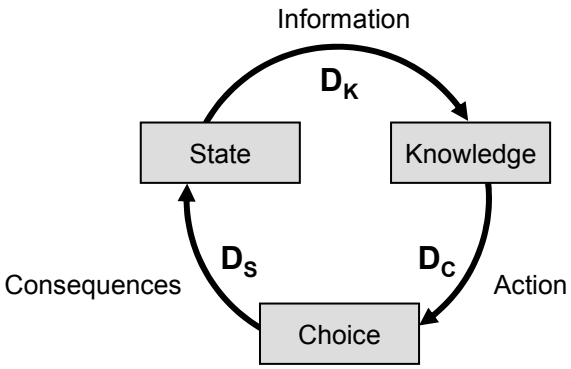


Figure 2-4 The information/action/consequences paradigm of system dynamics (adapted from [COYLE 1996, P. 4])

The modeling of system dynamics is usually done by influence diagrams that can contain different levels of information and detail. Simple influence diagrams only depict the system variables and the directed links between them (see Figure 2-6); and generally "+" and "-" signs are used for describing whether two connected variables change in the same or opposite directions. COYLE provides a comprehensive overview of the conventions for influence diagrams [COYLE 1996, P. 20FF]. Based on such models, a simulation approach can be implemented that allows for detailed analysis, interpretation, and optimization of systems. For example, VESTER & HESLER executed a comprehensive system dynamics model for ecology and planning in metropolitan areas from elementary influence diagrams to complex mathematical simulations [VESTER & HESLER 1980]. Both PALM and KARNOPP ET AL. present numerous examples for modeling the system dynamics of mechanical, electrical, and thermodynamic problems ([PALM 2005], [KARNOPP ET AL. 1990]).

System dynamics represent a powerful approach for modeling highly interrelated systems by standardized notation. Furthermore, the transfer of these models to mathematical descriptions allows a comprehensive analysis of system behavior. Whereas the basic modeling permits users to obtain a general system understanding, analyses require detailed functional information about implied elements and relations. Influence diagrams can lose clarity when the quantity of elements and dependencies increases [LINDEMANN 2007, P. 72]. Furthermore, analyses concerning the structural constellations in the modeled dependency networks do not comprise part of the system dynamics methodology.

Graph theory

The field of graph theory provides the fundamentals for many methods applied in product development, e.g., the critical path method or project scheduling [GROSS & YELLEN 2006, P. 493FF]. Numerous algorithmic problems can be traced back to the application of graph theory, which focuses on elements and their connection with dependencies. GROSS & YELLEN as well as BOLLOBÁS have compiled comprehensive information about the general topic, its basic theories, and generic applications ([GROSS & YELLEN 2006], [BOLLOBÁS 1990]). Graph theory allows system constellations to be characterized by, for example, their implied substructures (trees, cycles, strongly connected components, etc.) or their structural attributes (e.g., connectivity, coloring, or planarity of considered graphs). The possibility of structural characterization by substructures and attributes is an important basis for the thesis approach presented here. The application of structural criteria in the context of product development is detailed in Section 3.5.3 and in the Appendix.

A use case for the application of simple graphs is the mutual assignment of elements belonging to two different groups, as shown in Figure 2-5. Here, a company requires a certain number of jobs to be performed and each employee is suited for some of them. If each employee can perform one job at a time, it must be determined how many employees must be assigned to the jobs in order to perform as many as possible simultaneously. In Figure 2-5 the dashed lines represent the jobs an employee can perform, and the solid lines represent one possible assignment. GROSS & YELLEN present the algorithmic approach for solving this sort of problem by means of algorithmic graph theory [GROSS & YELLEN 2006, P. 533FF].

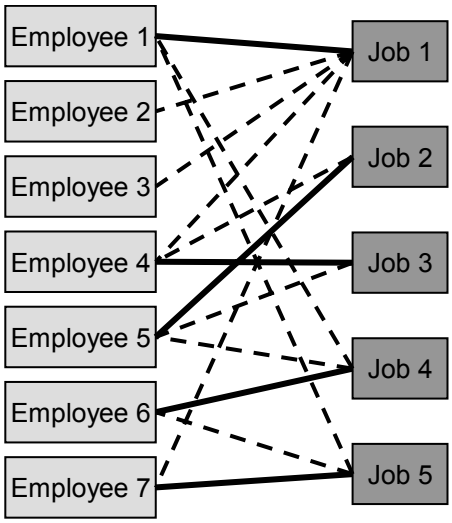


Figure 2-5 Assignment of employees to jobs (adapted from [GROSS & YELLEN 2006, P. 22])

The same authors also mention acquaintance networks or geographic adjacency of elements as common applications of simple non-directed graphs. Concerning directed graph (digraph) representations, Figure 2-6 depicts an application of a food web. The graph shows the "feeding relationships among the plant and animal species of an ecosystem […]. Each species in the system is represented by a vertex and a directed edge from vertex u to vertex v means

that the species corresponding to u feeds on the species corresponding to v" [GROSS & YELLEN 2006, P. 25]. In contrast to the simple graphs mentioned above, digraphs can present information about a flow direction and not merely the connectivity of two elements.

For the matrix methods being used in product development, graph theory in particular provides the mathematical basics for investigation into the dependencies between system elements. In fact, matrices only depict graphs in another representational form; most problems processed by matrix depictions can be efficiently solved by operations made available from graph theory ([BRUALDI & RYSER 1991, P. 23FF], [ANDRÁSFAI 1991, P. 133FF]). For example, self-energizing effects occurring between product components can be identified in graph theoretic structure analysis by cyclic connections of specific system elements. In order to benefit from the application of graph theory in complex product development, available possibilities must be introduced to a methodology, which can be applied by users without extensive mathematical investigation.

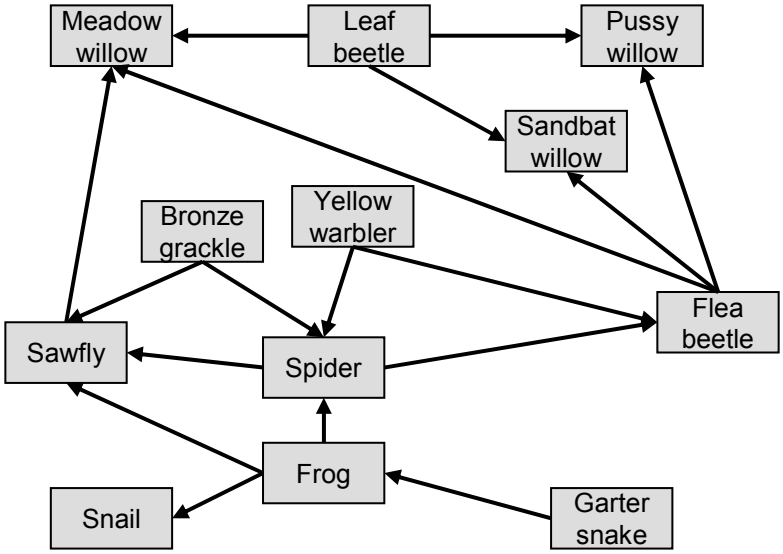


Figure 2-6 Digraph representing the food web in a willow forest (adapted from [GROSS & YELLEN 2006, P. 25])

2.3.2 Matrix-based approaches

In the early 1960s, several researchers proposed matrix-based methods for system modeling and analysis ([STEWARD 1962], [STEWARD 1981], [KEHAT & SHACHAM 1973], [WARFIELD 1973]). STEWARD used matrix-based techniques to analyze the structure of the system design process [STEWARD 1981]. These are similar to preceding diagrams, which had been used for project management before, for example, by FERNANDO and HAYES ([FERNANDO 1969], [HAYES 1969]). STEWARD represented information flows in binary form in matrices and developed algorithms for the realignment of elements in the matrix. During the last 20 years, these methods have increasingly attracted attention for managing the complexity of engineering systems and complex product development processes ([HAUSER & CLAUSING 1988], [KUSIAK & PARK 1990], [EPPINGER 1991], [YASSINE ET AL. 1999]). Thus, in 2003

YASSINE ET AL. stated that the idea of using matrix representations of a system for obtaining better understanding was not new [YASSINE ET AL. 2003].

The large variety of matrix-based methods in engineering can be classified by the quantity of the types of elements involved. Whereas some approaches focus on the representation and analysis in between elements of the same type (e.g., dependencies between product components), others consider linkages between two different types (e.g., dependencies between customer requirements and product functions). If relations within elements belonging to the same type are examined, the related matrices can be defined as intra-domain. Here, the expression "domain"[25] describes the type of element. Matrices combining different elements belonging to different domains are referred to as inter-domain matrices. For example, components and functions of a product can be considered as elements belonging to two different domains. More comprehensive approaches try to take into account even more domains by the sequential combination of inter- and intra-domain matrices. According to MAURER & LINDEMANN, such approaches are named Multiple-Domain Matrices [MAURER & LINDEMANN 2007].

Intra-domain matrices

The Dependency (called also Design) Structure Matrix (DSM) represents a commonly applied approach of an intra-domain matrix; the terms dependency structure matrix, dependency map, interaction matrix, incidence matrix, precedence matrix, and problem-solving matrix are also used in the literature [BROWNING 2001]. STEWARD ultimately defined the term "DSM" in a publication in 1981, when he applied matrix-based techniques to analyze the structure of a system's design [STEWARD 1981]. According to a description given by BROWNING, a DSM fulfils the following technical criteria [BROWNING 2001]:

A DSM is a square matrix, i.e., a matrix with an equal number of rows and columns. It provides systematic mapping of elements and their relationships. These elements can be, for example, physical product components, performance attributes, engineering requirements, or process tasks. The element names are placed down the side of the matrix as row headings and across the top as column headings in the same order. The diagonal elements of the matrix are not normally considered when describing the system (the self-reflexive edges of one element are not considered), so they are usually empty or blacked out ([WARFIELD 1973], [STEWARD 1981]). YASSINE ET AL. add that several types of DSM models can be built to capture the corresponding views of the product development process, i.e., process view, product view or organizational view [YASSINE ET AL. 2000]. DSMs, however, typically represent the elements of only one system at a time. Nevertheless, some applications integrate several dependency types in the same DSM ([PIMMLER & EPPINGER 1994], [JARRATT 2004, P. 128FF]). Such approaches can serve for information capturing, but computational analyses that are available for the common DSM can not be used on them.

Figure 2-7 shows an example of a DSM (adapted from [PIMMLER & EPPINGER 1994]) that is often mentioned in the related literature. The matrix elements represent physical components

[25] A detailed definition of the term "domain" in its application in the thesis approach is presented in Section 3.2.

of an automotive climate control system. The marked fields indicate interactions occurring between elements. PIMMLER & EPPINGER used the DSM to cluster the elements into chunks (also referred to as clusters), which helped them define the product modularization, and based on this, the structure of the development team.

As the sample DSM in Figure 2-7 only contains x–marks, it is called a binary DSM, i.e., a DSM populated with only zeros and ones, or equivalent symbols. These DSMs contain a minimum of information about the nature of the relationships (only the existence and the direction can be depicted). When STEWARD used the expression "Design Structure Matrix" for the first time, he referred to such a binary matrix [STEWARD 1981]. More detailed information on the relationships between elements is represented in numerical DSMs. These can contain a multitude of attributes, such as importance ratings or the probability of repetition [YASSINE ET AL. 2001]. Information can be added via the use of different marks in the cells, e.g., symbols, colors, or numerical rankings. Some approaches even represent multiple values in one DSM by subdividing the matrix cells [YASSINE 2004].

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Radiator	1		X														
Engine fan	2	X				X											
Heater core	3																X
Heater hoses	4																
Condenser	5		X				X		X								
Compressor	6					X			X	X							
Evaporator case	7																X
Evaporator core	8					X	X			X							X
Accumulator	9						X		X								
Refrigeration controls	10																
Air controls	11																
Sensors	12																
Command distribution	13																
Actuators	14																
Blower controller	15																X
Blower motor	16			X				X	X							X	

Figure 2-7 DSM of an automotive climate control system (adapted from [PIMMPLER & EPPINGER 1994])

Since STEWARD's research in the 1980s, the use of DSMs has been extended to many other types of system and design analysis, e.g., product development, project planning, project management, system engineering, and organization design [BRADY 2002]. YASSINE states that the literature is full of successful DSM implementations from various fields [YASSINE 2004]. There have been over 140 DSM-related articles in the past two decades [BROWNING 2001]. Some application scenarios are mentioned in the following:

- WHITNEY ET AL. used DSMs for capturing knowledge [WHITNEY ET AL 1999].

- In regards to process-oriented problems, EPPINGER ET AL. applied DSMs to activity dependencies and information flows [EPPINGER ET AL. 1994].
- YASSINE set up matrices for the transfer of documents and information [YASSINE 2004].
- YASSINE ET AL. supported the understanding of products and processes and demonstrated possibilities which could increase the efficiency of design processes and reduce development time [YASSINE ET AL. 2000].
- BROWNING & EPPINGER presented schedules and cost distributions for the execution of planned tasks [BROWNING & EPPINGER 2002].
- YASSINE provided possibilities for the analysis of systems and product architectures by DSMs [YASSINE 2004]. In addition, Browning indicated the suitability of DSM for the decomposition and integration of products [BROWNING 2001].
- WHITEFIELD ET AL. provided aid for developers by exploring the potential in design concepts by a systematic DSM approach [WHITEFIELD ET AL. 2001].
- A method for "change prediction" by DSM application is presented by CLARKSON ET AL., which tracks impact chains in product structures [CLARKSON ET AL. 2001]. Due to an initial adaptation to one single product component, eventually wide-spread consequences to a multitude of further components become obvious. As such consequences often remain unknown to developers, change prediction aids design improvement by reducing required rework.
- LINDEMANN ET AL. applied an approach similar to CLARKSON ET AL. for increasing possibilities of product customization [LINDEMANN ET AL. 2005]. Here, developers could identify product subsets that were suitable for customization measures, because linkages did not result in widespread change impact.

BROWNING states that the increasing use of DSMs in a variety of contexts is due to the "advantages of DSMs vis-à-vis alternative system representation and analysis techniques" [BROWNING 2001]. He provides a comprehensive and thorough review of the field of DSMs and their uses: "There are two main categories of DSMs: static and time-based. Static DSMs represent system elements existing simultaneously, such as components of a product's architecture or groups in an organization. Static DSMs are usually analyzed with clustering algorithms. In time-based DSMs, the ordering of the rows and columns indicates a flow through time: upstream activities in a process precede downstream activities, and terms like 'feed forward' and 'feedback' become meaningful when referring to interfaces. Time-based DSMs are typically analyzed using sequencing algorithms" [BROWNING 2001]. BROWNING further mentions four DSM applications: component-based or architecture DSMs, team-based or organizational DSMs, activity-based or schedule-DSMs, and parameter-based DSMs [BROWNING 2001]. Figure 2-8 shows the classification of DSMs and typically assigned analysis algorithms.

Besides the classification of DSMs by represented domains, the implied dependency meaning is also helpful for differentiation. Several authors provide lists of relevant dependency types for DSM modeling ([PIMMLER & EPPINGER 1994], [JARRATT 2004, P. 124FF]). Further useful

linkage types can be taken from general modeling approaches in engineering ([PAHL & BEITZ 1996, P. 29FF], [STEINMEIER 1999, P. 39FF], [EHRLENSPIEL 1995, P. 644FF]). MAURER & LINDEMANN highlight the importance of consistent dependency types in DSMs, as a mixture of different types impedes structural analysis and interpretation of structure characteristics [MAURER & LINDEMANN 2007].

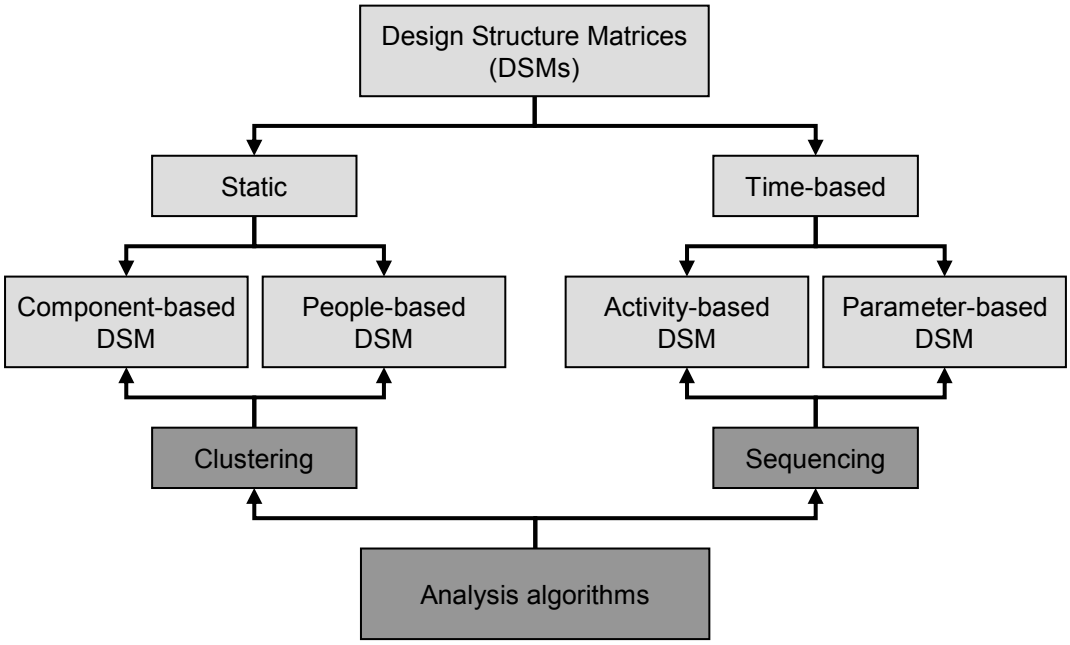


Figure 2-8 Classification of DSMs and analysis algorithms (adapted from [BROWNING 2001])

A set of analysis algorithms is available for the application of DSMs, which is discussed in greater depth in Section 3.4. All of these algorithms are based on the realignment of elements in the matrix in order to highlight structural characteristics. KUSIAK, YASSINE, and BROWNING provide comprehensive introductions to available possibilities ([KUSIAK 1999, P. 29FF], [YASSINE 2004], [BROWNING 2001]). KUSIAK contributes a sound step-by-step application of analysis algorithms like triangularization, feedback loop identification, and clustering that can be applied without the need for specific software support. However, if the quantity of elements in a system model goes beyond 30 elements, manual analysis becomes almost impossible. YU ET AL. present a genetic algorithm approach that uses the Minimal Description Length (MDL) principle for the identification and alignment of cluster and bus subsets in complex systems matrices [YU ET AL. 2005]. The authors provide applications of examples that are already too large for manual analysis but could still be handled by conventional clustering computation. Nevertheless, genetic approaches seem to be effective for the analysis of large and highly interlinked structures. Therefore, a genetic algorithm approach is presented in Section 3.6.2 that aims at the decrease of overall system complexity.

In general, DSMs are used to improve the design or the design process of a product. The aspect of knowledge capturing by a systematic process of matrix filling already contributes to this purpose. Future designers can benefit from the captured information about system connectivity, which will allow them to make designs faster and better. By appropriate

realignment of the element rows and columns, DSMs support the identification of structural subsets, which can be interpreted and provide methods of structural improvement.

Inter-domain matrices

Whereas intra-domain matrices consider relations within the same domain, inter-domain matrices link elements of two different domains. These sorts of linkages are widely used in design methodology and labeled with different names, such as "Cause and Effect Matrix" or "Interface Structure Matrix" ([ALLEN 2006, P. 121FF], [KUSIAK ET AL. 2006]). These approaches do not generally relate to the DSM methodology, i.e., available DSM techniques and algorithms are not systematically transferred to inter-domain matrices. In 2001, DANILOVIC & BÖRJESSON settled on the term "Domain Mapping Matrices" (DMMs) for a formal enhancement of the DSM methodology to inter-domain matrices. They presented studies on linkages between product architecture and organization as well as between systems and organization ([DANILOVIC & BÖRJESSON 2001A], [DANILOVIC & BÖRJESSON 2001B]). In 2003, DANILOVIC & SIGEMYR documented several methods that allowed the systematic analysis of DMMs [DANILOVIC & SIGEMYR 2003]. Subsequently, DANILOVIC & BROWNING described the DMM as the complement to known DSM approaches [DANILOVIC & BROWNING 2004].

Combined application of intra- and inter-domain matrices

Whereas single DSM matrices are commonly applied to different kind of structures, EPPINGER & SALMINEN describe the need for combining the three views of component, process, and organizational structures as a basis for successful product development [EPPINGER & SALMINEN 2001]. They stress the importance of comparison across the pattern types, because the "development organization is executing the development process, which is implementing the product architecture". EPPINGER & SALMINEN address the following questions by comparing DSMs:

- Does the organization properly execute the development process?
- Is the development process effectively implementing the product architecture?
- Are the architecture interactions driving the organizational communication?

The authors faced problems in the practical implementation of a systematic comparison, as appropriate analysis possibilities were not at hand. For this reason, they restricted their research to the comparison of one-to-one linkages between the three system views, i.e., they set up three DMMs for the inter-domain linkage [EPPINGER & SALMINEN 2001].

NASA developed two different extensions of DSM analysis to provide a more comprehensive system view of the project structure together with the technology choices [BRADY 2002]. Here, an interface DSM (see also Figure 2-9) maps the dependence within components and identifies component criticality and the probable resulting impact of adaptations on the project's operations. An additional DSM of the technology risk includes a technology risk factor for the components to help identify the patterns of high system risk. Without the use of an explicit inter-domain matrix, this combination of two DSMs has been used in two

spacecraft projects (NEAR and Mars Pathfinder) to expand the design and management teams' holistic views of the projects.

LINDEMANN applies DMM matrices for structuring problems in the systematic product development process [LINDEMANN 2007, P. 121F]. The matrix serves as a linkage between the objectives and attributes of possible solutions, i.e., this matrix correlates to a submatrix of the House of Quality, which will be more closely examined in the following section on multi-domain approaches (see also Figure 2-10) [HAUSER & CLAUSING 1988]. The Axiomatic Design approach also represents a multiple domain approach rather than a mapping between two types of elements. However, the methodology provides the sequential mapping between four different domains in DMMs [SUH 1988]. Starting with the domain of customer needs, the methodology maps functional requirements, physical design parameters, and process variables. SUH provides a sound mathematical basis for describing the transfer between the different domains. Interpretations for specific structural matrix constellations are also mentioned, e.g., the coupled and decoupled design [SUH 1988]. Even if the mathematical approach is well founded, it seems to be difficult to define and combine the required parameters correctly in practical applications of product development. WULF mentions that appropriate examples do not exist to illustrate possibilities for realizing mathematical linkages between functional requirements and design parameters in more complex systems [WULF 2002, P. 18].

DONG & WHITNEY introduce a systematic and mathematically founded approach to obtain a DSM from information in a Design Matrix (DM) [DONG & WHITNEY 2001]. Such DMs are applied in the Axiomatic Design approach mentioned before and relate to the requirements of design parameters (i.e., such matrices also represent DMMs) [SUH 1988]. DONG & WHITNEY explain that a DSM is appropriate for capturing, understanding, and managing "the interactions occurring in the system of the product and the system of the design team". However, "the traditional DSM method lacks the mechanism to construct the DSMs at early stages of the design". As "the relationship between the requirements and the design parameters is what engineers naturally think about", DMs are easier to create at early process stages, where the cost of changes is minimal [DONG & WHITNEY 2001].

The Interface Structure Matrix (ISM) represents a concept of modeling modularity based on the interaction among parts and interfaces [KUSIAK ET AL. 2006]. It is taken into account that conventional product modularization by clustering DSMs lacks appropriate possibilities of information acquisition. If designers can not clearly describe the relations between components, analytical methods of clustering fail due to wrong input data. As the authors state, "an ISM is easily derived from existing industrial databases such as a Bill of Material (BOM)". Figure 2-9 depicts the creation of an ISM from the system linkages as well as the optimized matrix obtained by realignment of the rows and columns. The applied cluster analysis and the resulting significance of the ISM are comparable to DSM approaches that can represent the same data by slightly different modeling. The advantage of the ISM compared to a DSM is the higher data quality, as required information often already exists in congruent form. This finding is also applied to the structure management approach presented in this thesis (see Section 3.2).

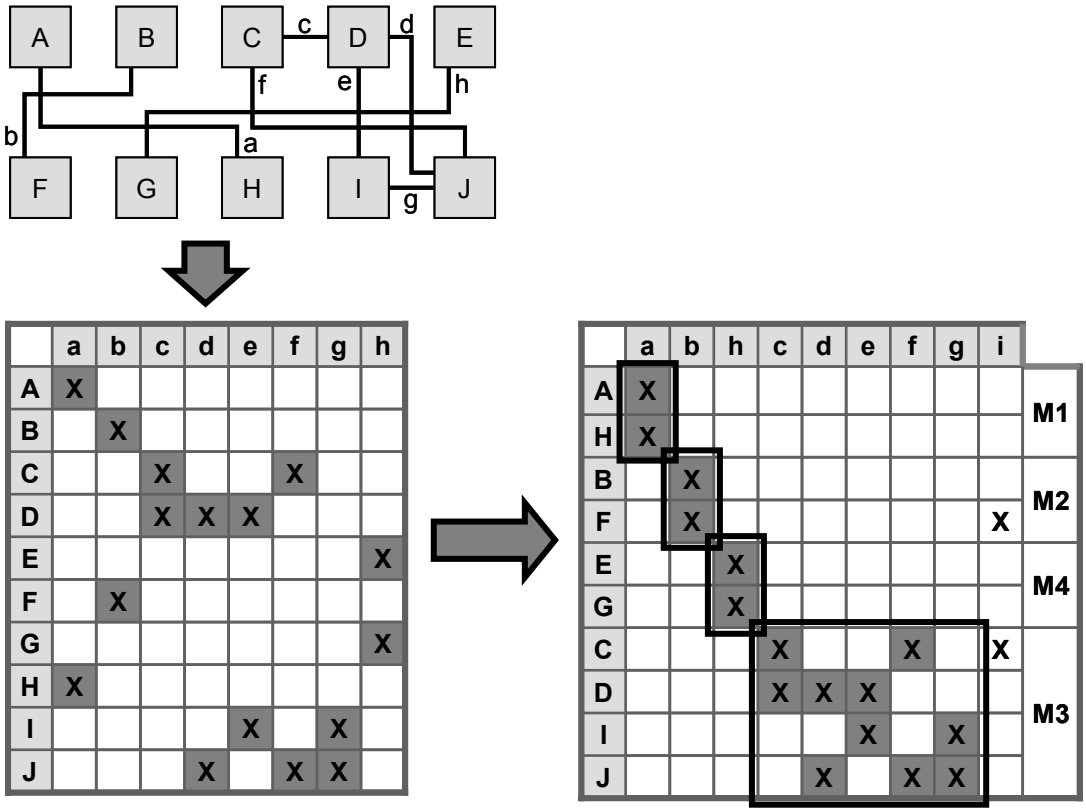


Figure 2-9 Application of the Interface Structure Matrix (ISM) (adapted from [KUSIAK ET AL. 2006])

Multiple-Domain Matrices

Multiple-Domain Matrices (MDMs) represent a consequent combinatorial enhancement of DSMs and DMMs. The term MDM has only been mentioned by MAURER & LINDEMANN who point out its close relation to DSM and DMM approaches [MAURER & LINDEMANN 2007]. For example, HERDER-DORNEICH introduces a general layout of such a combination of several domains, but does not provide methodological procedures for their further analysis [HERDER-DORNEICH 1992, P. 284F]. In design methodology several practical applications exist, but algorithmic analyses are rarely available.

A well-established MDM is given by the House of Quality (HoQ), which comprises part of the method of the Quality Function Deployment (QFD) ([HAUSER & CLAUSING 1988], [AKAO 1992]). Figure 2-10 shows the fundamental layout of the HoQ and indicates the integrated DSM and DMMs. As can be seen, the triangular "roof" of the HoQ forms a DSM, as only relations between elements of the same element type (technical requirements) are depicted in the matrix fields. The dependencies are non-directional, i.e., only the existence and in some cases the intensity of the influence between technical requirements can be acquired. Although this matrix is not square, it fulfills all further requirements on a (non-directional) DSM. In the application process of the HoQ, the "roof"-DSM contains information about possible requirement conflicts or positive correlations. Additional DSM analysis like clustering seems

promising, as, for example, positively correlating requirements could be modularized; however, such analyses have not been applied so far. Besides the "roof"-DSM, the HoQ is comprised of three DMMs that link customer requirements and technical requirements, customer requirements and criteria, and technical evaluation and technical requirements. For all three DMMs, numerical dependency values are usually captured, i.e., the intensity of existing relations is noted in the matrices. A possible analytical enhancement would be the matrix alignment by element similarity, as provided by MCCORMICK ET AL. and DANILOVIC & BROWNING ([MCCORMICK ET AL. 1972], [DANILOVIC & BROWNING 2004]). This would permit grouping customer requirements, for example, due to their similar linkage to technical requirements.

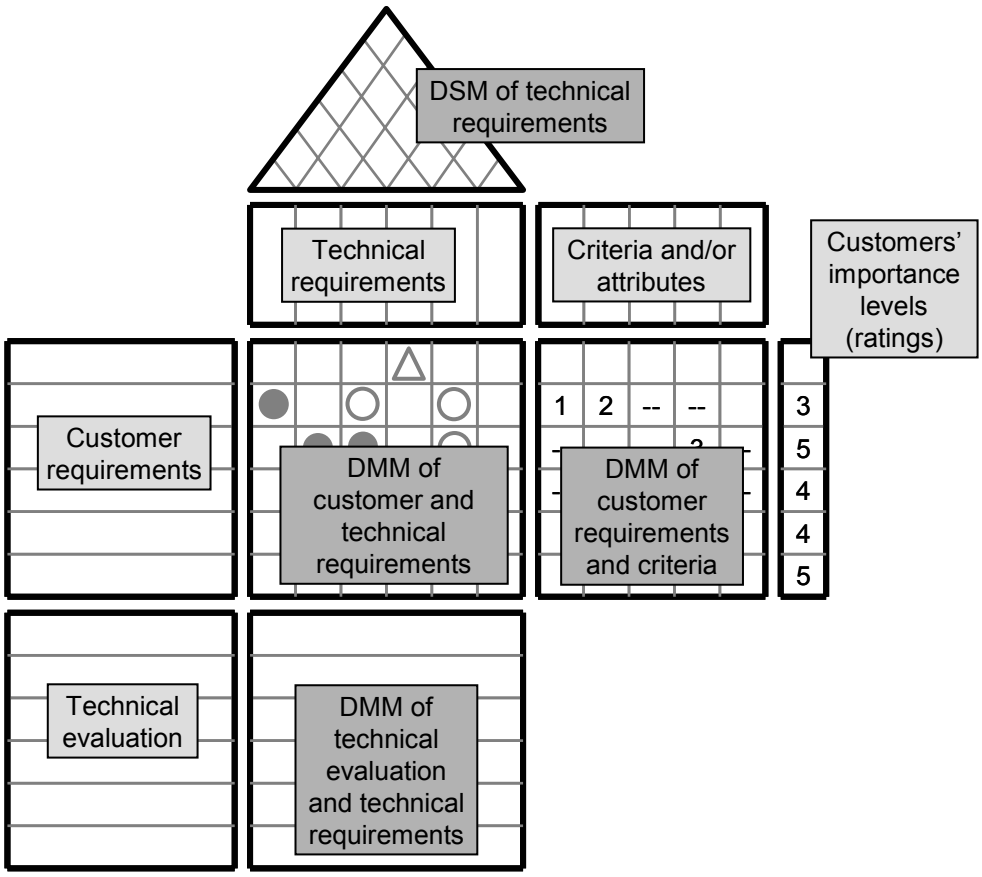


Figure 2-10 The House of Quality and implied DMM structures (adapted from [USHER ET AL. 1998, P. 3])

In addition to the general layout of the House of Quality, HAUSER & CLAUSING provide a sequence of "linked houses" that "convey the customer's voice through to manufacturing" [HAUSER & CLAUSING 1988]. Starting from the combination of customer requirements and technical characteristics, this sequence is continued on to component deployment, process planning, and production planning. As explained before, identical algorithmic analyses can be useful for a methodical enhancement.

BONGULIELMI ET AL. introduce the "K- & V-Matrix" for managing different types of configuration knowledge [BONGULIELMI ET AL. 2001]. An example of matrix alignment can

be seen in Figure 2-11. The "K- & V-Matrix"[26] comprises two DSMs (named V-Matrices) describing the internal dependencies of the technical and the customer view of a product. Both matrices are linked in the K-Matrix, which maps the technical system view with the customer view, and therefore represents a DMM. PULS ET AL. describe a software implementation that allows for practical applications [PULS ET AL. 2002]. In addition, PULS describes the possibility of generating one of the three matrices by computing available information in the two other ones [PULS 2002, P. 89FF]. However, required mathematical procedures are not detailed. PULS further indicates that analysis results in the form of a positive or negative correlation of different views and configurations that can be obtained by applying the K- & V-Matrix, but without mathematical descriptions that allow practical implementation.

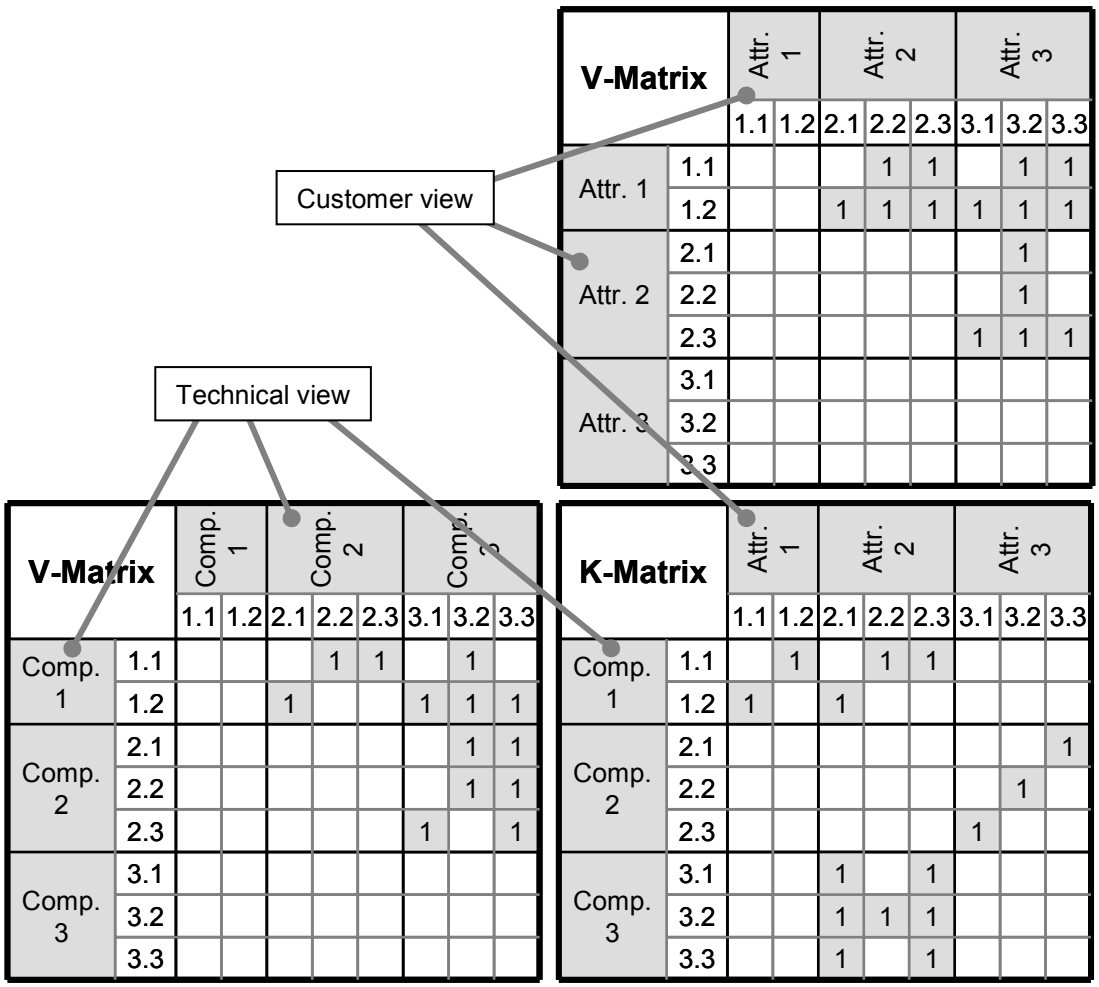


Figure 2-11 General composition of the K- & V-matrix (adapted from [PULS 2002, P. 90])

[26] The denotation "K" is derived from the German term "Konfiguration", which means configuration; "V" is derived from the German term "Verträglichkeit" and means compatibility.

DANILOVIC & BÖRJESSON apply a MDM to the management of a multi-project environment [DANILOVIC & BÖRJESSON 2001A]. They generate separate DSMs on the tactical and operational level and provide links to transform strategic business decisions into a series of projects. The findings "show that project structure and the prevailing basic organization cannot be regarded as contradictory and mutually exclusive. Instead of competing with each other, these dual structures need to find a balance of focus". DANILOVIC & BÖRJESSON present several projects in DSMs and connect them by DMMs, but only mutually link single project tasks without using advanced computational methods [DANILOVIC & BÖRJESSON 2001A]. In subsequent research work, DANILOVIC & SIGEMYR present a software tool, which supports the multi-project management that is based on the preceding research [DANILOVIC & SIGEMYR 2003]. One unique advantage of the approach is the possibility of dealing with an asymmetric DSM, i.e., a DSM with the same elements but aligned in different order on the two axes.

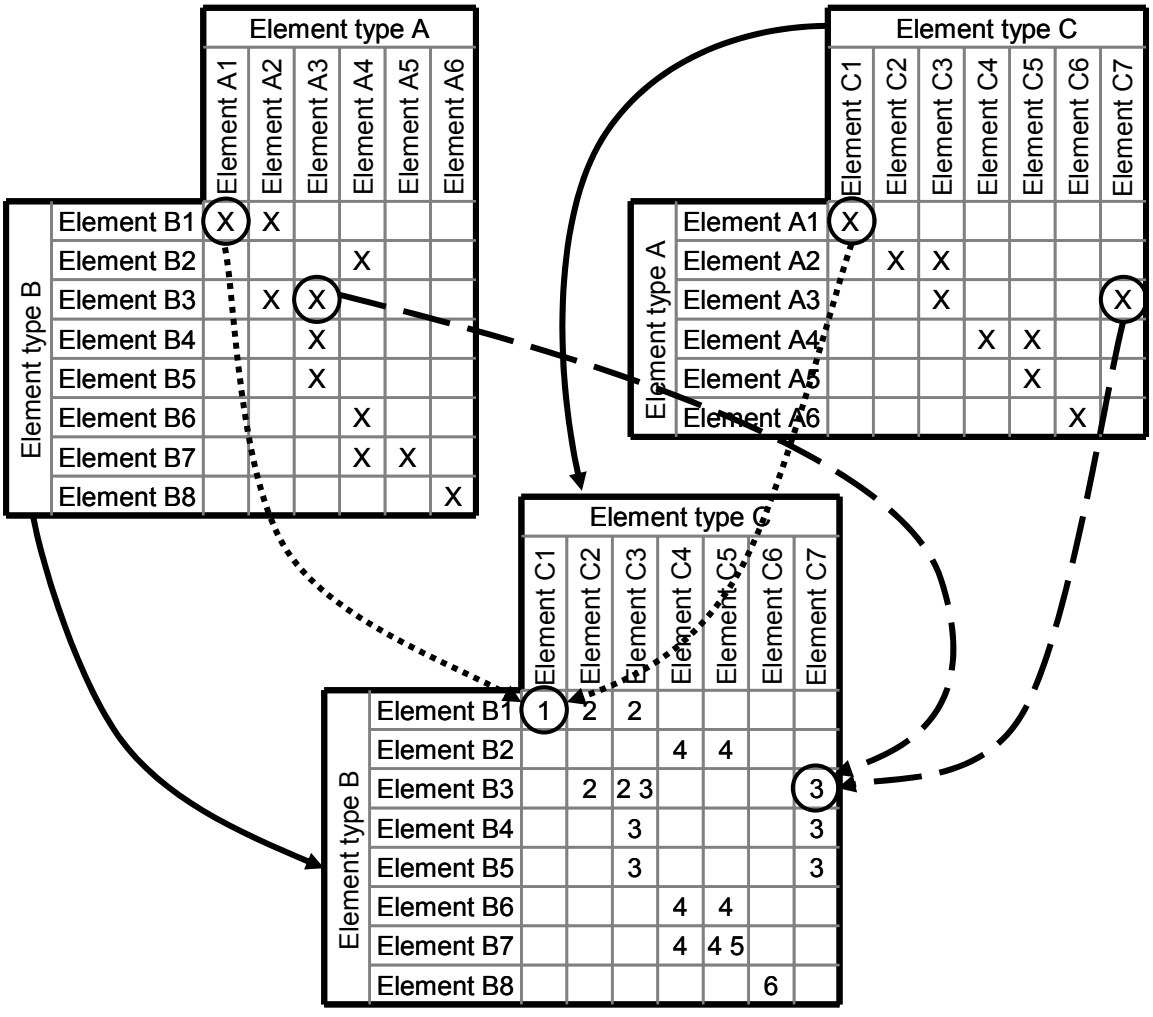


Figure 2-12 Connectivity maps (according to [YASSINE ET AL. 2003])

YASSINE ET AL. present a multiple domain approach called "connectivity maps" [YASSINE ET AL. 2003]. In regards to the decomposition approaches provided by DSM methods, they state that "decomposition helps in containing the technical complexity of the design; however, it

increases its managerial complexity. The synthesis of the different elements (or subsystems) into a final product (or system) requires the identification and understanding of the inter-relationships among the different elements". Further they note that it is the linking within and between domains that "makes the product and its development complex, and makes capturing relationships between affiliated elements necessary if the product engineering is to be handled efficiently and effectively, and if the product is to be robust and successful." The methodical approach of connectivity maps is shown in Figure 2-12 and allows the computation of a DMM on the basis of two other DMMs, if these overlap in one of their domains. It can be seen in the figure that knowledge about dependencies between the domains A and B as well as between A and C allow for the computation of indirect dependencies between the domains B and C. In the resulting DMM, the numbers indicated in the matrix cells specify the elements in domain A that provide the indirect linking of elements between domain B and domain C. YASSINE ET AL. state that "essentially, it is a combination of two relationship maps into a single N x M matrix". The mathematical fundamentals and possible interpretations of the results are described in the research work. So far, the approach is unique among known multiple-domain approaches, as it computes a DMM instead of a DSM. This seems to be promising for future structure analysis. However, only a few analysis approaches are available that allow the consideration of DMMs so far (e.g., the similarity approach previously mentioned [DANILOVIC & BROWNING 2004]).

2.4 Summary

It is a fact that complexity poses a challenge in many areas of product development and that several problems hinder developers from an intuitive complexity management in practice. Strategies with a focus on complexity have been developed that can be classified in different groups concerning the specific objective. Thus, the acquisition of complex systems represents an important issue, as the quality and correctness of available data determines all further measures. Nevertheless, only a few approaches possess possibilities for the systematic capturing of data and are mainly designed for automated import of measurable data. Concerning non-quantified or non-documented data that are based on the experience of experts, hardly any systematic approaches are documented that can be applied in practice. Furthermore, structure acquisitions only cover small amounts of elements and need to be improved in order to enlarge the scope of application to realistic use cases.

Several approaches are available for the evaluation of complexity but need to be improved, especially when applied to highly interconnected systems. Criteria for evaluation of variant management are available but have not been applied to other aspects of product development so far. A similar situation exists for approaches which focus on the avoidance and management of complex systems, as well as variant management.

Whereas several strategies of complexity management rate complexity as a negative attribute, some authors argue that complexity is advantageous in specific situations, e.g., to permit enhanced system flexibility. The trend towards increasing product customization particularly demands enhanced product flexibility, as customer-oriented products and requirements for a short time to market represent competitive factors for enterprises. For such applications, the reduction of complexity and thus flexibility may even decrease competitiveness in the market.

Other authors indicate that the capability to control complexity implies a competence that can not be easily imitated by competitors. This competence allows for fast product adaptations with less iterations, and therefore less required resources in case of engineering changes. If, however, the complex connectivity within systems remains uncontrolled, single adaptations can impact many other components, and the resulting measures are often unknown. Even if several authors mention the importance of controlled engineering changes, so far no comprehensive methodology has been available that can be applied to practical development situations.

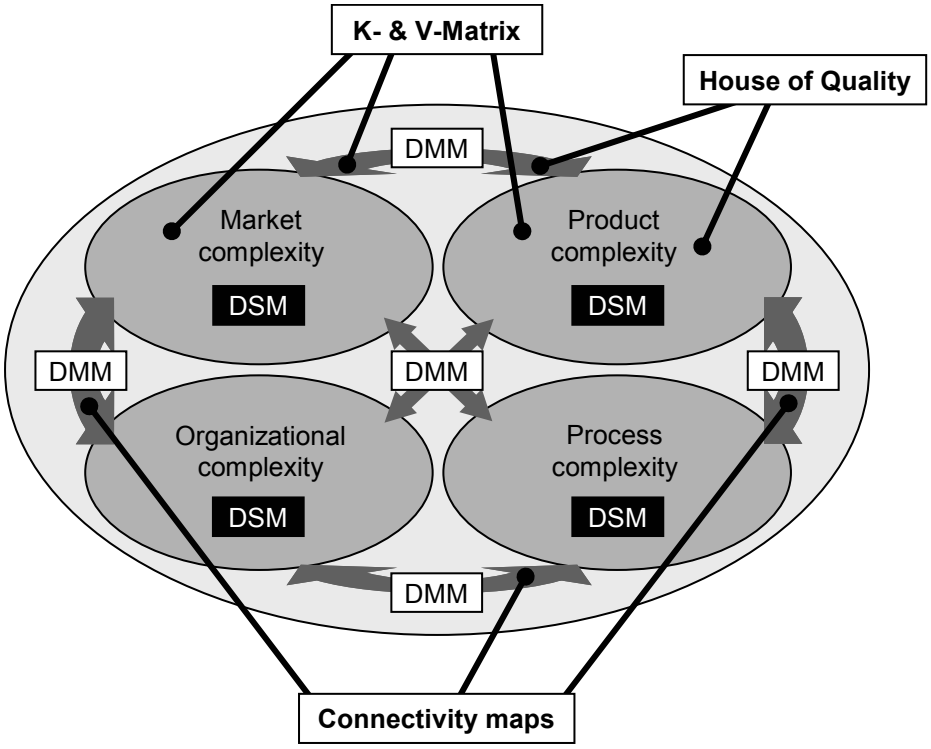


Figure 2-13 Assignment of matrix-based methods to the aspects of complexity in product development

For the interaction with complex systems, many different visualization techniques and computational approaches have been applied; however, matrix-based methods have been commonly used in product development. Starting from the modeling of relations within one domain, e.g., within product components or process steps, the mapping between different domains has been acquired by a multitude of development methods. Several authors note the importance of more comprehensive methods that take different domains into account simultaneously. Although a few approaches are documented in the literature, a lack of appropriate analysis methods exists. Figure 2-13 depicts the matrix approaches considered in the context of the aspects of complexity already shown (see Figure 1-1). Whereas DSM approaches focus on the connectivity within one domain, DMM approaches consider the linkages between two domains. The multi-domain approaches mentioned combine several DSMs or DMMs. The K- & V-matrix connects elements of the market complexity (implied in a DSM) with elements of the product complexity (implied in a DSM) by a DMM. The House

of Quality possesses a DSM of technical components that can be assigned to an aspect of product complexity and enhances this by several DMMs that introduce customer requirements. Enhancements of the basic House of Quality are well documented in the literature, e.g., design, marketing, finance, and production which are simultaneously turned into engineering decision-making processes [USHER ET AL. 1998, P. 2]. Connectivity maps combine two DMMs in order to compute a third one; the approach is provided on a generic level and therefore focuses on the combination of user-defined aspects. The only constraint of application is that the two applied DMMs must include one identical domain.

In summary, many approaches face the challenge of managing complexity in product development, but only a few focus directly on the structures implied in the system considered. As complexity often emerges from the connectivity between different aspects and disciplines, a new approach to complexity management is required which handles various element and dependency types in combination with their structural connectivity.

From the classification of approaches in Figure 2-13 it can be stated that matrix-based methods do not provide possibilities for a holistic consideration of the aspects of complexity in product development. Therefore, the present thesis elaborates upon such an approach in the following chapter. Most of the established matrix-based methods apply algorithms that are basically founded on graph theory, but restricted to methods that can be visualized in matrix depiction. These methods have to be transferred and enhanced when considering a complexity that is characterized by the integration of multiple domains.

3. Structure management for controlling complexity

The consideration of complexity in product development previously discussed showed that managing complex structural dependencies represents both a major challenge and a success factor for enterprises. Complexity not only appears in specific sections of product development, but it is also a concomitant feature throughout the entire process in all relevant process domains and their linkages. Methods applied to complexity management generally focus on specific development tasks (e.g., identification of conflicting requirements) or particular objectives (e.g., product modularization). Therefore, they do not fulfill the requirements of a comprehensive structure consideration that controls and even benefits from existing complexity. In this chapter a methodical approach to structure management is presented that closes the gap between the methods presented and allows the comprehensive analysis, control, and optimization of structure-based complexity in product development. The approach consists of a fundamental system and suggests several known methods as special use cases. The general outline will be presented first, followed by detailed discussions on the particular steps of the approach.

3.1 Structure management approach

Typically, product structures that are taken from development processes only depict a portion of the complex reality developers have to act in (e.g., the component decomposition of a product). If specific aspects are considered independently from their holistic embedding, such isolated views may lead to measures that are harmful in the entire context. In order to avoid the disadvantages of considering incomplete parts of the entire system, some approaches to product structures include multiple aspects, e.g., general impact networks [LINDEMANN 2007, P. 72FF]; consequently, the complexity of these models increases because of a higher quantity of elements considered and more intense connectivity. Besides the extensive requirements for visualization, one further disadvantage of such approaches is their minimal suitability for computational analysis (as will be shown in Section 3.2). The depiction of multiple aspects results in the simultaneous existence of different element and dependency types, which does not allow the application of algorithms or interpretations of structural characteristics (see also Figure 3-2 and the attendant explication).

The content of complex systems can not be entirely represented on a detailed level; the information required has to be selected according to the use case. In practical application users may fail to grasp the graph or matrix in its whole, if hundreds of elements are depicted simultaneously in matrix or graph representations. A survey on the user friendliness of matrix structures showed that small matrices containing only a few elements can easily become too complex to be handled by developers [MAURER ET AL. 2006]. BROWNING even states that "individuals may have difficulty building DSMs with more than ten elements" [BROWNING 2001]. Large system structures are often visualized in graph representations, where the positioning of elements can be freely chosen; however, the elements' relative location to each other is important in order to perceive the representation. The positioning of elements in

graphs can be understood according to specific aspects, e.g., following a horizontal time line; unfortunately, some other aspects are often neglected by such alignments, such as the hierarchical decomposition of elements. Appropriate possibilities of representing complex product structures will be discussed in Section 3.4.

The previous comments point out two basic methods for avoiding a system modeling that is too complex. On the one hand, subsets or system views may be extracted from the entire system for closer consideration. Therefore, the complexity of a model can be reduced, but system analysis may not be useful, because the embedding of the subset has not been considered. On the other hand, a holistic system structure can be processed if it is created on a more abstract level. This also reduces the quantity of system elements considered, but unfortunately, the analyses attained may result in more general findings.

The existing possibilities shown above argue for the provision of a systematic approach for managing complex structures in product development. The procedure that is presented here is based on established problem-solving approaches as provided by DAENZER & HUBER, EHRLENSPIEL, and LINDEMANN from an engineering point of view and by ULRICH & PROBST from a holistic cybernetics point of view ([DAENZER & HUBER 1999, P. 96], [EHRLENSPIEL 1995, P. 79FF], [LINDEMANN 2007, P. 45FF], [ULRICH & PROBST 2001, P. 112FF]). It enhances the systematic structure analysis process for DSMs, as proposed by YASSINE ET AL. [YASSINE ET AL. 1999]. The outline is given in Figure 3-1 and comprises five general steps leading from the initial problem to the desired solution. The starting point can be a handling problem or a design problem, both resulting from a system's complexity.

A handling problem can occur if a product or a development process already exists, but the demand for adaptation generates problems of system controllability. Even if a specific adaptation request is clearly specified and does not appear to be complex, system dependencies may lead to numerous subsequent adaptations (change propagation). Such after-effects are often unknown, and therefore can not be anticipated – leading to time and resource shortages when attempting to accommodate the desired adaptation. In order to minimize handling problems, product developers require a manual that provides relevant system information about impact chains or the consequences of adaptations. The structure management approach allows for the acquisition and easy-to-access representation of a system's internal dependencies, and therefore allows the evaluation and management of adaptation requests. If the structure of system dependencies is present, users can define optimization potential, e.g., reducing the effort of changing for future system adaptations.

The motivation for applying the structure management approach can also be a design problem. This concerns the new development of a system (e.g., a product) and represents the logical enhancement of the structure handling mentioned before: in product design, system structures generally result as the passive consequence of the design process; i.e., developers create a product that possesses a certain structure, even if that structure is not directly considered during the design. Sometimes, methods such as functional modeling are applied, which support the creation of a product structure compatible to the functional requirements; however, in most use cases, product structure requirements are not considered as part of the design process. If systems become too complex and must be adapted, the structure handling mentioned before is required. However, if product development already includes a pro-active

structure development, clear design improvements can be realized; i.e., less iterations occur in comparison to passively emerged structures and change propagations are locally restricted.

One of the central reasons for applying structure management is that handling and development problems can not clearly be separated: if the system handling is improved, optimization potentials may be detected that may lead to a new or improved system design. Likewise, a system design that includes pro-active structure development also requires the handling of complex structures. For this reason, both initial problems are linked, as shown in Figure 3-1.

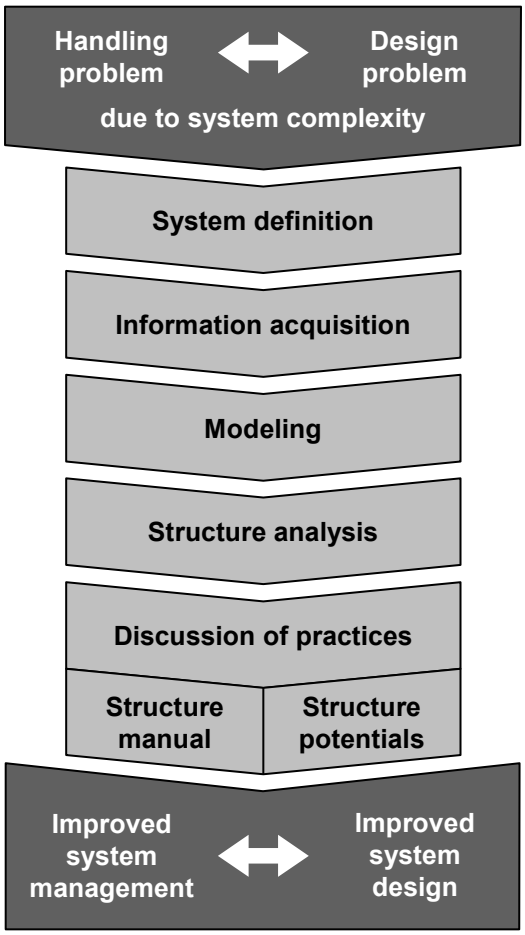


Figure 3-1 General structure management approach

Once the type of initial problem and required solution is determined, the system definition represents the first step of the structure management approach. The system definition serves to specify the scope of consideration and provides the basis for all further steps of the approach. Tasks include the identification of required domains, the determination of system elements and their level of detail, and the linking dependency types, i.e., mapping logics between specific domains. Generally, the objective of the system definition is to identify required information and set up the basis for the systematic extraction of knowledge and user access. For most analyses and optimization scenarios, a system definition that coherently

depicts information on multiple domains is compulsory. Even if the structure management approach in Figure 3-1 is depicted only as a sequential flow, iterative execution of the system definition can become necessary in practical application. If, for example, more information on the system layout becomes known later in the process, the suitability of the system definition must be reviewed. A detailed description of the system definition step is given in Section 3.2.

Data may be acquired from the existing data bases, modeling tools, or by interviews. The selected method depends on the specific use case[27] and availability of information. Interviews require time-consuming workshops with experts; however, if the system in question is only comprised of implicitly known experience knowledge or data that has not been documented thus far, interviews provide the only possibility of capturing required information. The main challenge of such an acquisition is to guarantee a high quality of resulting data. This calls for the application of appropriate methods, which help to consider all relevant information and to avoid effects resulting from symptoms of fatigue. In contrast to this, information acquisition by extraction from available data bases or software tools seems to be easier, because it can at least be partly executed automatically; however, some requirements must be fulfilled in order to successfully interact with large amounts of data. The process step of data acquisition is further detailed in Section 3.3.

If required data are on hand, they must be modeled in order to allow user interaction. Requirements for preparation may include a comprehensible visualization, possibilities of navigation in the structure, user-defined system views, or highlighting specific structural characteristics. Numerous techniques are available and chosen depending on the objective. In Section 3.4 an appropriate selection is proposed that fulfills the requirements for the approach presented.

The step of the structure analysis (detailed in Section 3.5) serves to identify the system's characteristics, specifically by applying the algorithmic graph theory. Several established analysis approaches only apply specific parts of the mathematical fundamentals, e.g., matrix depictions that are used for the identification of criteria by the realignment of rows and columns. The approach presented here integrates the available techniques in a general methodology. Some analyses focus on the characterization of the entire structure and its behavior, whereas others focus on the structural embedding of single elements and dependencies. The application of structure analyses provides the basis for further system optimization. Section 3.5 explains the procedure of structure analysis, and the definitions of analysis criteria are presented in the Appendix.

Finally, the discussion of practices applies the results from system analysis in order to provide solutions for initial handling or design problems. In this step of the approach, a structure manual can be created that provides a better understanding of the system in question. Furthermore, possible structural optimization measures and possible ways to improve the structural design can be composed. Methods for executing the discussion of practices are described in Section 3.6.

[27] Two different use cases for information acquisition are presented in this chapter in Figure 3-20.

In particular, the modeling, analysis and optimization tasks of the approach presented require software support in order to apply them effectively. Requirements for the implementation are detailed in Section 3.7.

3.2 System definition

In order to solve a problem resulting from structural complexity, the scope and implied aspects of the system model in question must be clearly defined. That means that system domains, the level of detail of elements, and the meaning of dependencies considered must be determined. Based on this, the requirements for the subsequent step of information acquisition can then be derived. A clear system definition is necessary for the visual and computational model representation and is indispensable for the application of analysis methods.

Several system definitions are applied to methods in product development focusing on structural constellations. For example, impact networks represent systems by elements and dependencies without further classification. Consistency matrices differentiate dependencies between elements by two classes, consistency or inconsistency, or introduce different levels of consistency on an ordinal scale. An even more specific system definition is applied using relation-oriented functional modeling [LINDEMANN 2007, P. 119F]; two different kinds of elements (two types of functions) are linked by three different dependency types resulting in a high density of informational content of the depicted models. However, a system model of only one type of element and one type of dependency is often better suited for structural investigations. The advantages of this model compared to those loaded with more information are that the resulting networks can be easily understood by users, since informational overload by a multitude of domains and dependency types can easily overextend perceptivity. Furthermore, the application of computational analysis methods for the determination of significant structural constellations is only useful when applied to a dependency network that includes dependencies of identical meaning. For example, if elements represent product components, and dependencies describe their mutual change impact, a feedback loop can be interpreted as a reflexive effect of the element that was initially adapted. If a structure consists of different element and dependency types, such an interpretation can not always be applied. Figure 3-2 depicts both use cases in a generic example: if component 1 is initially adapted, the left feedback loop indicates a closed impact chain between the three components and can result in self-energizing or self-impeding effects. In the loop shown at the right side of Figure 3-2 this interpretation is invalid, as an adaptation to component 1 does not necessarily propagate by the different loop elements and dependencies.

The DSM methodology makes use of homogeneous networks and provides the possibility of analysis for better insight into them. Depending on the specific use case, the appropriate domain (e.g., components or functions) and dependency meaning (e.g., change impact) is considered, and specific methods are available for static or time-based structure content. Unfortunately, some disadvantages of system models exist in practice if these models are only comprised of one domain and one dependency type. People initially tend to intuitively model systems with multiple element and dependency types. Often this seems to be the only way to depict complex situations. System complexity often emerges from the interaction of different domains or disciplines. Developers are able to control the dependencies within the domain of

their own development scope; however, external linkages (e.g., required software adaptations due to mechanical changes) remain unconsidered. This unknown connectivity to external aspects can provoke unwanted system behavior. Therefore, it seems inappropriate to build a model of only one domain. Such methods typically select one aspect of the complex system and neglect problematic inter-domain dependencies.

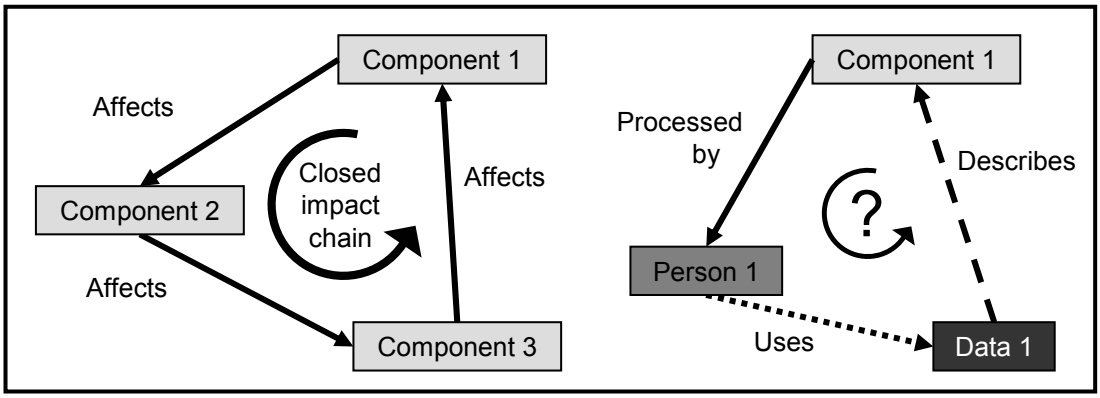


Figure 3-2 Feedback loops in system structures

The creation of complex systems requires modeling multiple domains and dependency types. In contrast, user comprehension and application of analysis methods requires systems of only one domain and one dependency type. Consequently, it is necessary to acquire different system aspects separately and to allow for the case-specific derivation of individual system views from the comprehensive model.

3.2.1 The construction of a Multiple-Domain Matrix

Figure 3-3 shows an example of a network of eleven elements that belong to three different domains. To better distinguish between them, the elements possess circular, triangular, and square forms and are denoted by Latin characters, Greek characters, and Arabic numbers. The elements are linked by four dependency types, indicated by different lines. The domains could represent components, people, and data, for example. Dependency types could depict change impact, information flow, geometric, and thermal dependencies. The system, however, is not easily comprehensible even if it only consists of a few elements. Obviously, this is due to its high density of informational content. As explained before, analyses of the structure in Figure 3-3 seem to be impractical, as most assignable constellations (subsets) would comprise several dependency and element types, which hardly allow for interpretation.

An examination of the system in Figure 3-3 initially requires its systematization. Figure 3-4 shows the extraction of network information that can be further processed by the DSM methodology: only the subset of dependencies connecting elements in between the same domain can be depicted in the corresponding square matrices. In the following, such dependencies will be addressed as intra-domain dependencies. For the network used as an example, four different DSMs are required to systematically capture all included intra-domain dependencies. Separate matrices have to be built for every domain, and two matrices are

required for the green (triangular) one. As two different dependency types occur in this domain, these have to be transferred to different matrices. The extracted DSM-conforming subsets then allow for the application of analysis algorithms (presented in Section 3.5).

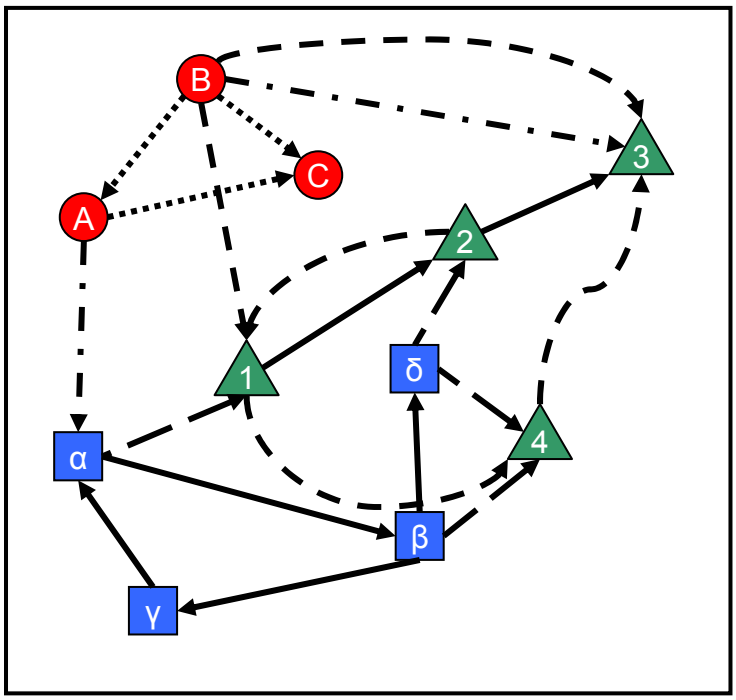


Figure 3-3 Elements and dependencies in a system model

The dependencies connecting elements of different domains can be systematized like the preceding extraction of DSMs. Such dependencies will be addressed as inter-domain dependencies in the following and can be transferred into Domain Mapping Matrices (DMMs). Figure 3-5 shows this process for the network example. Although only three combinations exist for linking between the domains, four DMMs are required for storing the network information that belongs to the domains. Just as between the red (circular) and the green (triangular) domain two dependency types occur, dependencies must be transferred to different DMMs.

Whereas several methods for DSM analysis are known in literature, DMM analysis has rarely been applied thus far. Approaches focus on clustering or alignment of elements due to their mutual similarity (see Section 3.5 and the Appendix). However, a DMM is often used for connecting two DSMs, as in the case of the K&V-matrix [BONGULIELMI ET AL. 2001]. In Figure 3-6, the overall matrix combination of DSMs and DMMs is depicted for the network example. This comprehensive matrix system will be denoted as Multiple-Domain Matrix (MDM) and includes all DSMs and DMMs that have been extracted before (see Figure 3-4 and Figure 3-5) and classified by the three domains. The MDM possesses all features of a common DSM; in fact, it represents a DSM on a higher level of abstraction. If the domains are considered as single elements, the DMMs represent the dependencies between these elements. The DSMs are then located on the matrix diagonal and represent self-reflexive

dependencies that are not considered in typical DSMs. Consequently, the MDM in Figure 3-6 can be understood as a DSM consisting of three elements (representing the domains) that are connected by three dependencies (representing the DMMs).

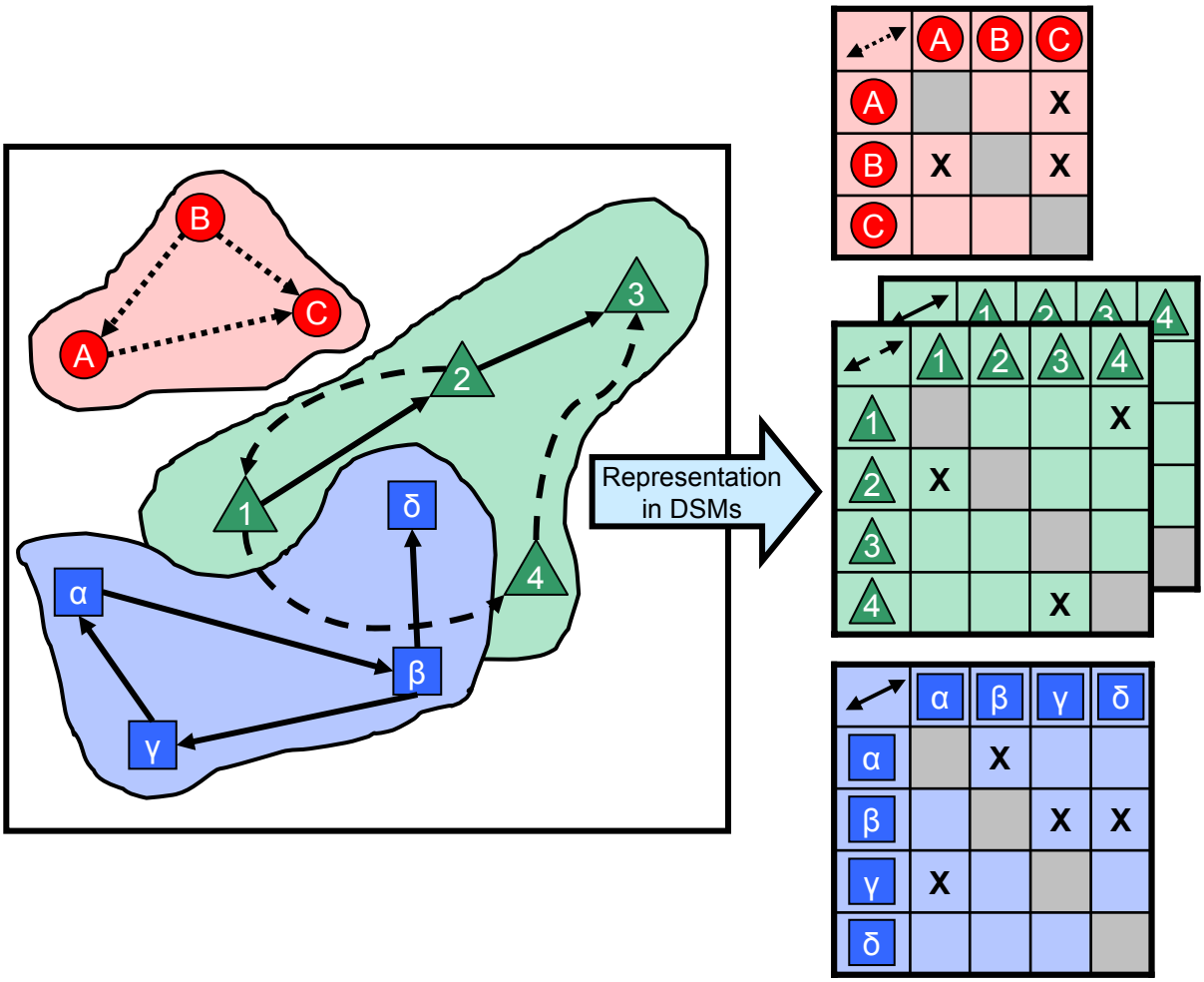


Figure 3-4 Representation of intra-domain dependencies in DSMs

The MDM allows the systematic modeling of networks comprising different domains and dependency types. Although MDM models are not restricted to isolated views of single domain and dependency types, computational analysis can still be performed, as subsets of homogeneous content are represented separately in the implied DMMs and DSMs. Thus, required information can be selected and even combined, depending on the specific use case.

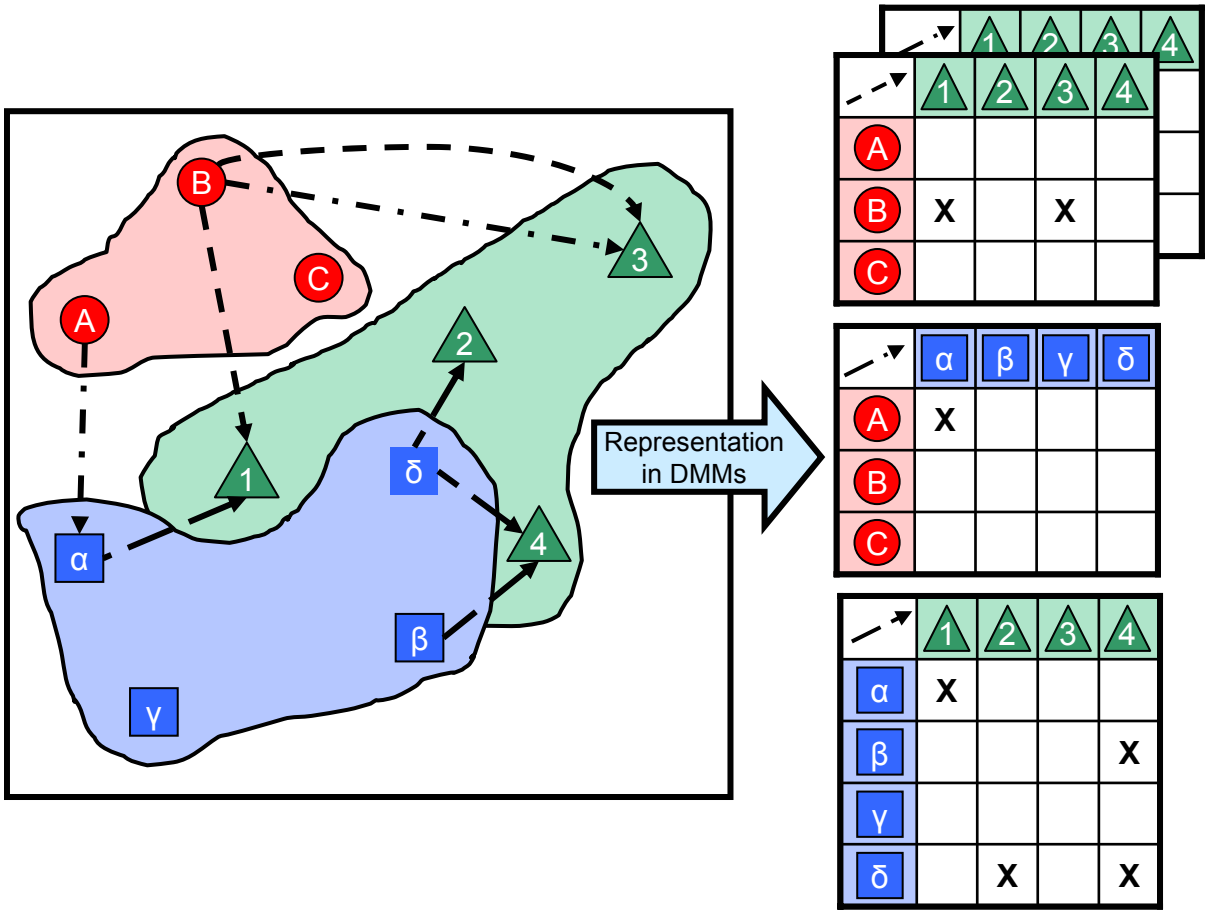


Figure 3-5 Representation of inter-domain dependencies in DMMs

It must be noted that the MDM is generally not a two-dimensional matrix system. As it is suggested in the example in Figure 3-6, different matrices can exist for the same intra- or inter-domain linking. This means that two different dependency types are depicted that exist within the same combination of domains in parallel. In Figure 3-6, the MDM consists of two DSMs of the green (triangular) domain and two DMMs linking the red (circular) to the green (triangular) domain. For reasons of consistent representation, the matrices are stored separately, resulting in a three-dimensional alignment in the depiction. Three fields of the MDM in Figure 3-6 are empty, i.e., no dependencies exist for this specific domain linking. For executing structural analysis, it is not required that a MDM is completely filled; rather, possible computations depend on available DSMs and DMMs.

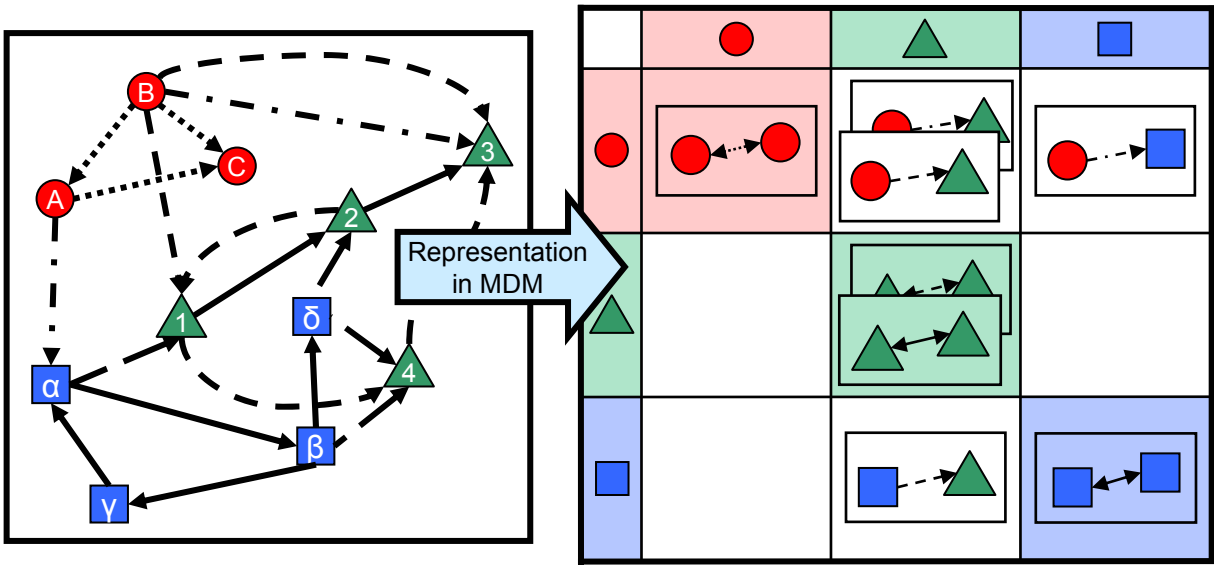


Figure 3-6 Representation of dependencies in a MDM

Although a MDM is generally comparable to known DSMs, items included have to be clearly defined. The following figures sequentially introduce these items, and Figure 3-10 depicts them all in a comprehensive context. The applied MDM example is identical to the one in the preceding figures. However, from Figure 3-9 on, an additionally derived DSM is created. Subsequently, Table 3-1 lists all applied terms with a short explication.

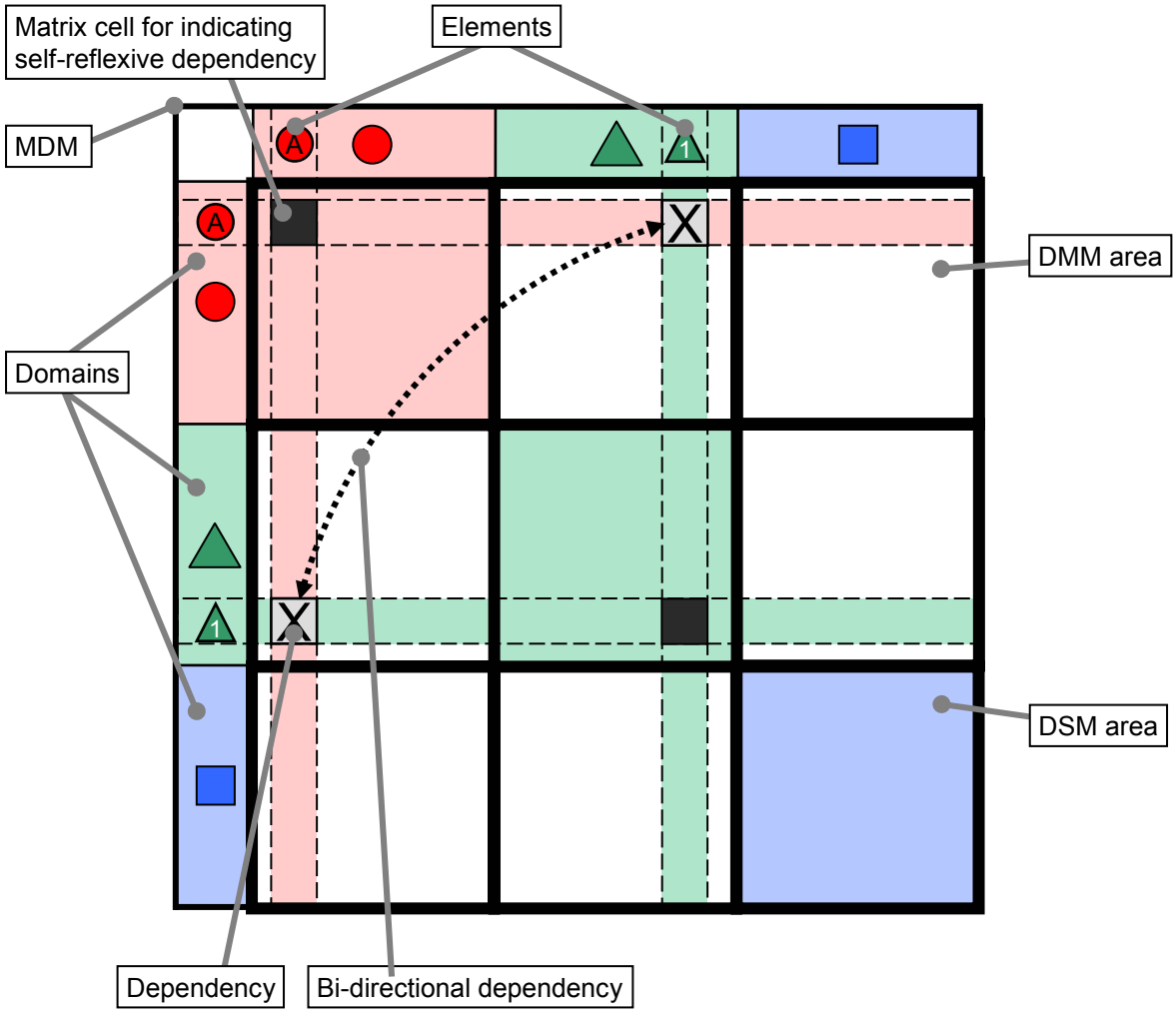


Figure 3-7 Composition of the MDM 1

Figure 3-7 depicts the basic elements of a MDM, which is comprised in this example of three domains (indicated by the red, green, and blue coloring and the circular, triangular, and square forms of elements). However, a MDM consists of at least two and up to a theoretically infinite number of domains depending on the specific use case. The domains represent the classification of the elements in groups. Examples of domains are people or documents, and single elements represent specific instances from these groups. As a MDM is a square matrix, the order of domains as well as the order of elements within the domains is identical on both axes. The areas of intersection of rows and columns belonging to the same domains represent DSM areas, as all dependencies that can be indicated in this area connect elements within the same domain. Consequently, intersections of rows and columns belonging to different domains represent DMM areas. Elements are depicted in the row and column heads, and every element indicates a row and a column of the matrix. In the intersection of both, self-reflexive dependencies, for example, could be noted, but typically these sorts of dependencies are not applied. Every other intersection of a row and a column can represent a dependency between the elements by implementing a cross, for example. A MDM can depict the direction

of the dependency between two elements, and the interpretation follows the rule “row affects column”. A bi-directional dependency between two elements is indicated, if two matrix cells are filled, which are laterally reversed to the matrix diagonal.

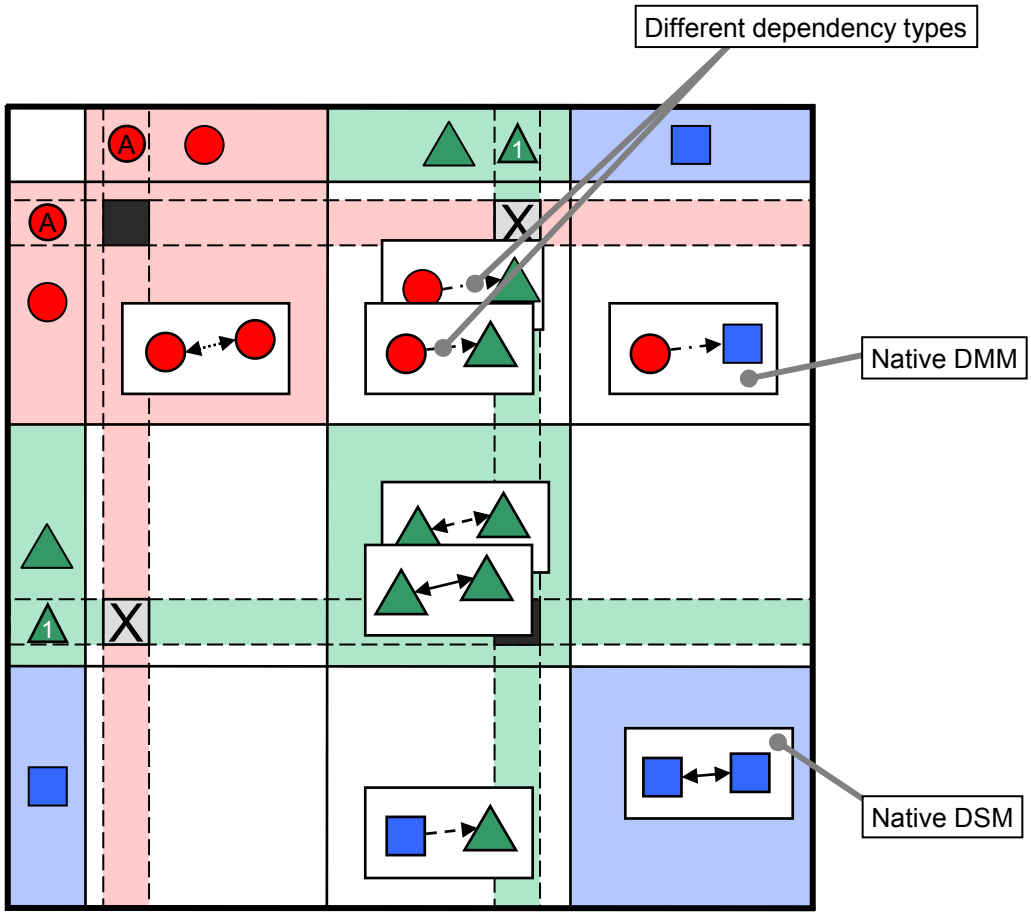


Figure 3-8 Composition of the MDM 2

Figure 3-8 introduces data to the different areas of the MDM. DSMs and DMMS are mentioned as “native”, if they result from data acquisition. In contrast, DSMs (and DMMs) can be derived from other data by computation. This will be explained in the following figures. A dependency type represents the meaning of a dependency, e.g., “change impact” (between components) or “information flow” (between process steps or people). If within the same DSM or DMM area, different dependency types occur, they have to be stored in separate matrices. This allows later access to specifically required dependency information.

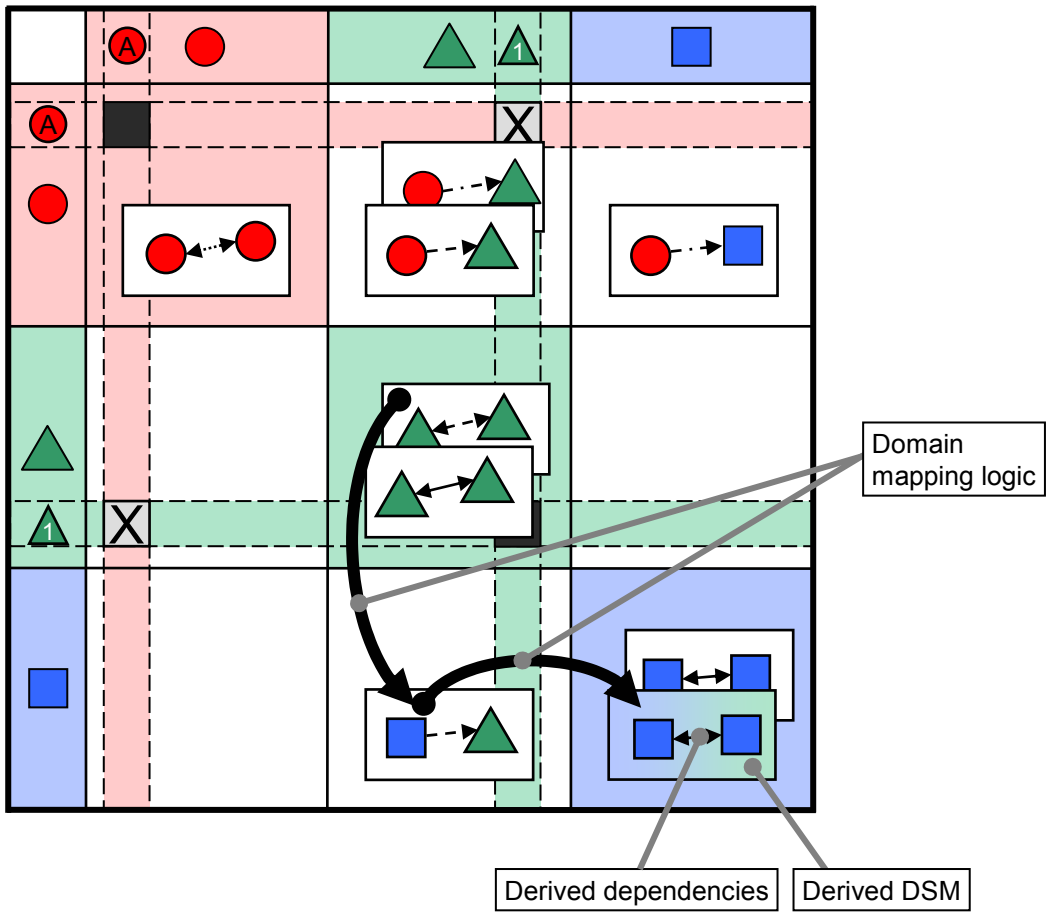


Figure 3-9 Composition of the MDM 3

Figure 3-9 clarifies the creation of a derived DSM. As mentioned above, data in a derived DSM do not emerge from information acquisition but from computation by means of further data. Whereas the logical and mathematical procedures will be explained later, here the relevant terms are introduced. A derived DSM contains derived dependencies that emerge from the executed computation and connect between the elements of the considered domain. In Figure 3-9 the derived dependencies result from the combination of one other DSM and one DMM. Such a combination of DSMs and DMMs for computing the dependencies of derived DSMs is denoted as domain mapping logic and will be the subject of further discussions beginning with Figure 3-13. Generally, derived DMMs also can be created by applying suitable domain mapping logics [YASSINE ET AL. 2003]. However, due to a lack of available analyses and possible interpretations, such derived matrices are not further considered in the approach presented.

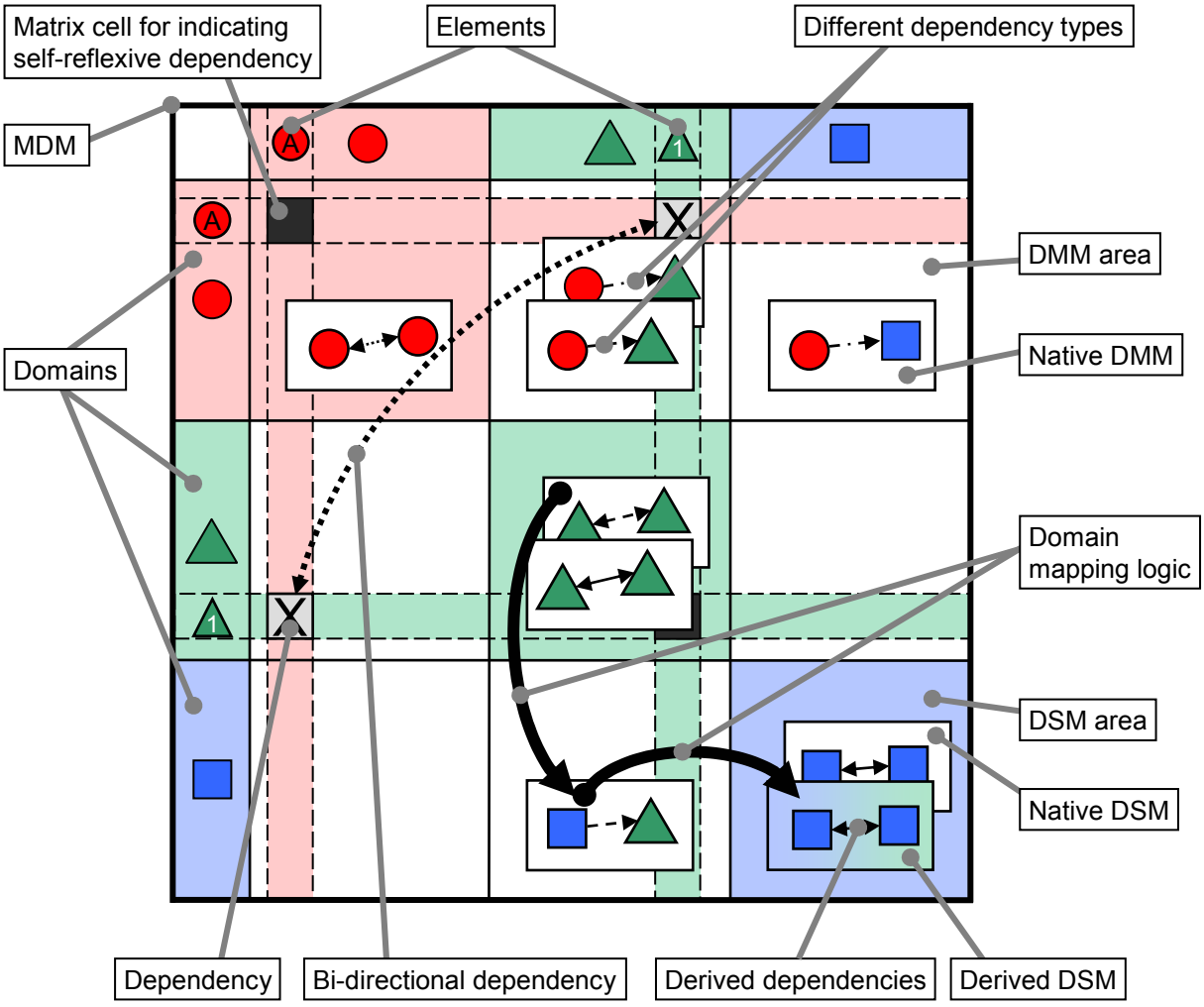


Figure 3-10 Composition of the MDM 4

All items of a MDM are presented together in Figure 3-10. With this representation form, product developers possess a solid basis for systematic depiction and consideration of comprehensive information for complex interrelated structures. As different kind of elements and dependencies are not merged, computational analyses can be applied depending on the specific requirements. In the following, the possibilities of creating derived dependencies by domain mapping logics will be discussed in further detail. The corresponding computational requirements are presented in Section 3.5.1.

Table 3-1 Item definitions for the MDM

Component	Explanation
Bi-directional dependencies	*Bi-directional dependencies* connect two *elements* in both directions and are depicted by two filled matrix cells aligned symmetrically to the matrix diagonal.
Dependency	A *dependency* is the relation between two *elements* represented by a symbol or a number in a matrix cell and is normally directed. The *elements* connected by the *dependency* are depicted in the column and row head belonging to the matrix cell.
Dependency type	The *dependency type* describes the meaning of a *dependency*. Different *dependency types* can even exist between the same *elements* (and between the same *domains*). Different *dependency types* are stored in separate matrices. Examples of *dependency types* are “change impact” or “information flow”.
Derived dependency	A *derived dependency* represents an indirect *dependency* of two *elements* due to an indirectly existing *dependency* path between both; *derived dependencies* in a *DSM* can be determined by *dependency* information stored in one *DMM*, two *DMMs*, or the combination of one or two *DMMs* with a further *DSM.*
Derived dependency logic	The *derived dependency logic* is the *dependency type* of a *derived DSM* resulting from the *domain mapping logic* and the meaning of *domains* and *dependency types* applied for the computation.
Derived DSM	A *derived DSM* represents *elements* and *dependencies* computed by information taken from further (*native* or *derived*) *DSMs* and *DMMs*.
DMM	Abbreviation for Domain Mapping Matrix; a *DMM* is a matrix connecting *elements* belonging to two different *domains*. *Elements* of the first *domain* are aligned along the vertical axis; elements of the second *domain* are aligned on the horizontal axis.
Domains	*Domains* represent the primary classification of *elements* in a *MDM*. Examples of *domains* can be people or documents.
Domain mapping logic	The *domain mapping logic* describes the computation scheme for creating a *derived DSM* out of *DMMs* and *DSMs*.
DSM	Abbreviation for Design Structure Matrix; a *DSM* is a square matrix comprising the same *elements* in identical order on both axes. *Elements* in a *DSM* all belong to the same *domain*. In the matrix cells *dependencies* between *elements* can be indicated by symbols or numbers, but only one *dependency type* is applied per *DSM*. The direction of *dependencies* can be defined as “row *elements* influencing column *elements”* or vice versa.
Elements	*Elements* represent single nodes of the system structure and are aligned symmetrically at both axes of the *MDM*. Every element belongs to one *domain*.
Matrix cell for indicating self-reflexive dependency	A *self-reflexive dependency* connects one *element* with itself. Such *dependencies* can be stored in the matrix cells of the diagonal in *DSMs*, but are typically not considered in *DSM* methodology.

Table 3-1 Component definitions for the MDM (continued)

Component	Explanation
MDM	Abbreviation for Multiple-Domain Matrix; the *MDM* is a square matrix comparable to a *DSM* containing system elements in identical order on both axes. In contrast to a *DSM* different types of system *elements* are included and grouped in *domains*; the *MDM* can be subdivided into *DSMs* and *DMMs* according to the inherent *domains*.
Native DMM	A *native DMM* represents *elements* from two *domains* and the *dependencies* between them that are directly compiled from information acquisition.
Native DSM	A *native DSM* represents *elements* within one *domain* and the *dependencies* between them that are directly compiled from information acquisition.

3.2.2 Creation of derived Design Structure Matrices from Multiple-Domain Matrices

As mentioned before, system structures comprising elements belonging to only one domain and connected by only one dependency type are best suited for the user's comprehension and application of structural analysis. Such structures are represented by the DSM subsets in a MDM. Figure 3-11 shows the process of computing a derived DSM by means of two other MDM subsets. On the upper left side, the elements and the dependencies (thin lines) represent a subset of the same structure applied in the preceding figures. At the right side of the figure, the two MDM subsets are symbolized, which can be extracted from the graph representation. Additionally, the DSM that can be computed by the two subsets is depicted. The computational procedure answers the following question:

How do elements of the blue (square) domain depend on each other because of their dependencies to elements of the green (triangular) domain and further dependencies in between elements of the green (triangular) domain?

This abstract formulation can be understood in the context of an example, such as when the blue (square) domain represents people and the green (triangular) domain represents product components:

How are people linked to each other because they work on (different) components, which possess mutual change impact?

This question is of interest in development processes, for example, because the derived network indicates developers who have had to cooperate since they work on interrelated components. If one developer implements changes to his dedicated component, the second developer has to react, because his component needs to be adapted, too.

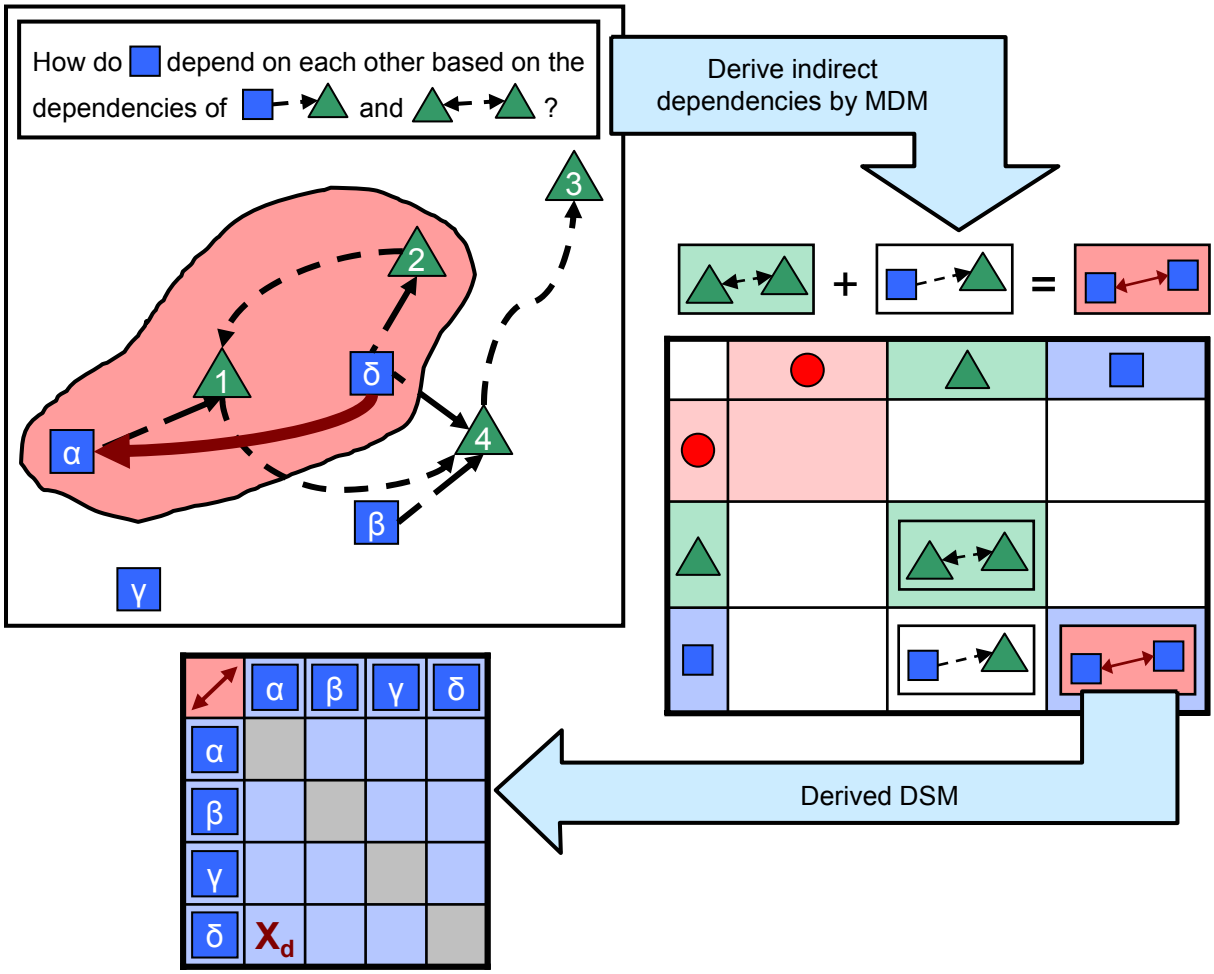


Figure 3-11 Process of deriving indirect dependencies

In Figure 3-11, the derived DSM of the blue (square) domain represents the subset of the MDM that has to be computed for answering the question above. A DSM representing the intra-domain linking of the blue (square) domain has already been extracted as a native DSM from the original data (see Figure 3-4). However, the derived DSM of the blue (square) domain implies a different dependency type. The mathematical approach for determining the requested DSM will be described in Section 3.5, and the logical procedure will be introduced here.

Elements α and δ of the blue (square) domain both possess a link to elements of the green (triangular) domain (elements 1 and 2); additionally, element 2 possesses an intra-domain link to element 1; consequently, an intra-domain linking of the blue (square) domain can be created from element α to element δ that represents the indirect dependency chain in a condensed form. In Figure 3-11, this dependency is depicted by a dark red (thick) arrow.

Although the deduction of such dependencies is easy in small graph representations, it becomes almost impossible if networks become larger. A possible alternative is the systematic dependency deduction by use of the MDM, as it is visualized in a three-step

instruction in Figure 3-12, which only shows the green (triangular) and the blue (square) domain from the preceding example:

Step A First, elements of the blue (square) domain are considered in regards to their possible dependencies on elements of the green (triangular) domain. Such dependencies can be found in Figure 3-12 in the lower left DMM, which links the blue (square) domain to the green (triangular) one. In practice, the rows containing the blue (square) elements in question are followed into the DMM subset until a filled matrix cell is detected.

Next, it must be determined if the newly found elements of the green (triangular) domain are mutually interrelated. This information can be found in the DSM of the green domain.

Step B The columns belonging to the dependencies identified in step A are followed upward into the DSM of the green domain.

Step C The rows corresponding to the actual columns (i.e., containing the same elements in the row and column heads in the DSM of the green domain) are browsed for dependencies in their intersection with the columns mentioned.

Within the green domain in Figure 3-12, only element 2 links to element 1 and not vice versa. Consequently, only a derived dependency from element δ to element α results in the derived DSM of the blue domain, indicated by X_d.

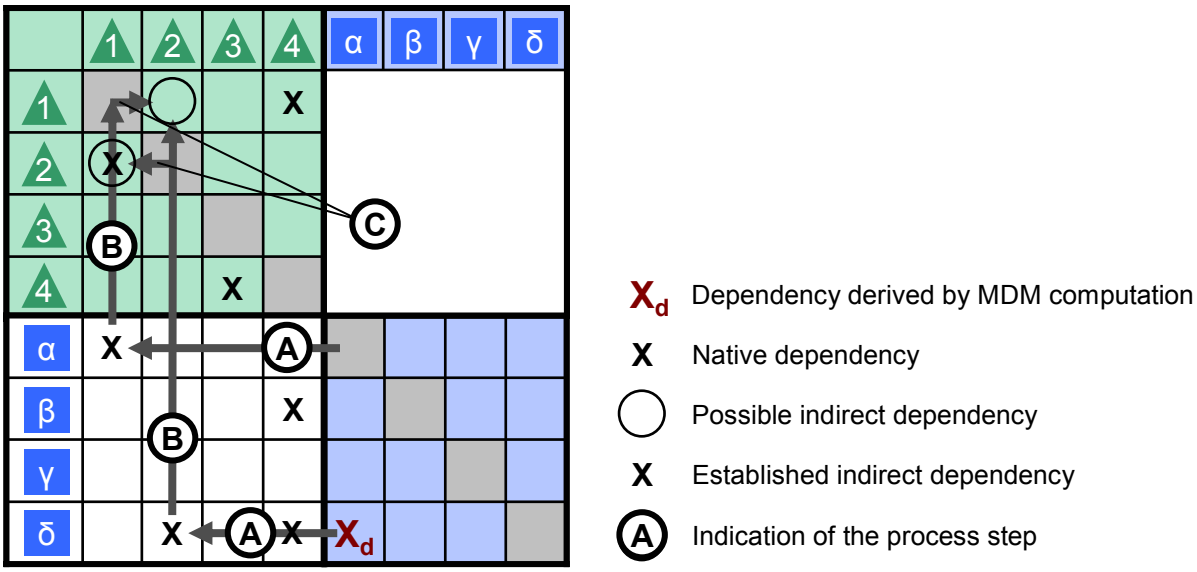


Figure 3-12 Process of deriving indirect dependencies by visual matrix examinations

Six different possibilities exist for computing an intra-domain network from dependencies between two different domains, if dependencies possess a direction. Generally, undirected or bi-directional dependencies could also be represented by DSMs and DMMs, and the additional consideration of these dependency types would increase significantly the quantity of derivable DSMs. However, the logic procedure remains the same. An overview of the six basic computational logics can be seen in Figure 3-13.

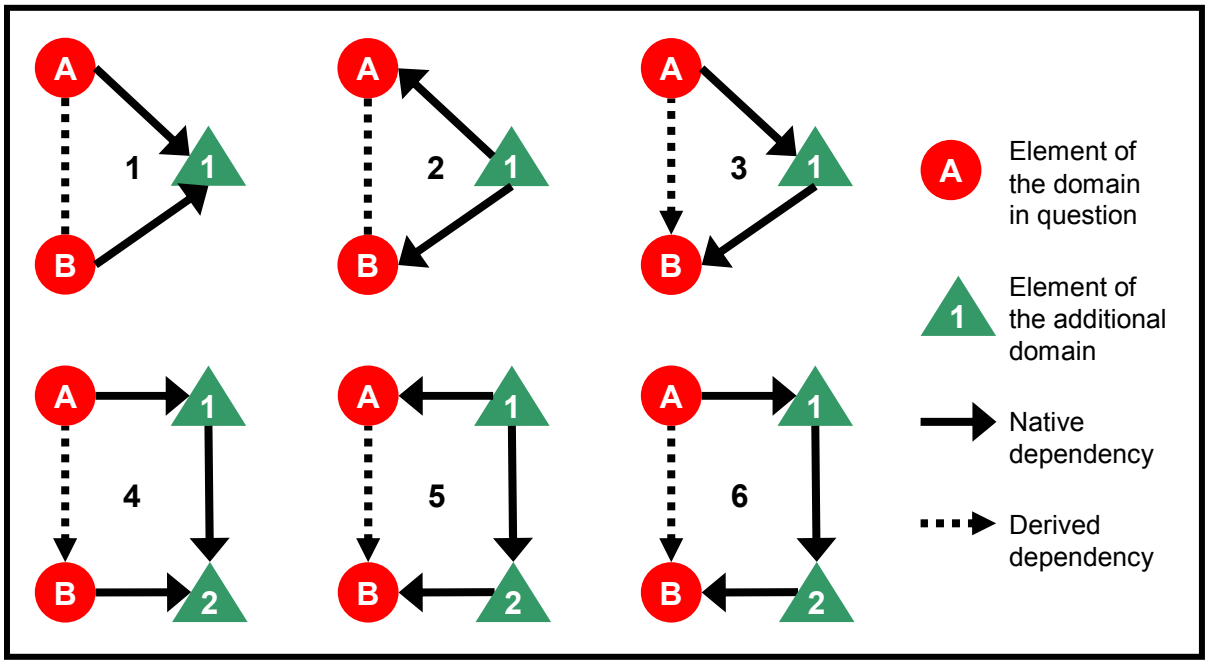


Figure 3-13 Domain mapping logics for deriving DSMs

For a detailed explanation of all six use cases, the following example is applied. Dependencies between people (red circles) are derived by existing dependencies of people to documents (green triangles) and in between the documents. This represents a typical scenario in product development, where designers depend on other designers due to mutual data transfer. Of course, the intra-domain network of people could be directly compiled without taking a second domain into consideration. Such a people-based DSM could, for example, depict the existing team structure that is based on the people's communication dependencies. However, it is often easier to take a detour by applying a second domain for the following reason: if dependencies between people are to be directly determined, people have to be asked about their connectivity to other people because of mutual data delivery or requirements. This question would often be difficult to answer correctly; for example, a person may create a document but does not know about the specific person who requires it next. Therefore, he will save the document to a server without information about its further usage. In such cases, the resulting network quality could be poor.

If the documents are introduced as a second domain, the questions become easier to answer. First, people have to declare which documents they provide (without the need for specifying a target person); this information is captured in a DMM linking people to documents. Secondly, people have to declare the documents they require to be provided for their work (without the need for specifying the source). This information is captured in a DMM, which links documents to people. The initially required dependencies between people can then be derived from acquired information and represented in the people-based DSM. In the following, all six possibilities for deriving a DSM are considered in detail.

Case 1 (Single DMM construction 1): A DSM can be created from the information content taken from one DMM only. Figure 3-14 shows the logic dependencies in graph as well as in matrix alignment. Whereas the upper part of the figure depicts the generic model, the lower part applies the use case example mentioned before. According to the depiction, dependencies between people can be deduced from people's linking to documents. That means two people become linked in the derived network because they work on or access the same document. The generated DSM does not provide a linking direction, as implied informational content only specifies the identical access to the same document. However, according to the definition, every filled matrix cell of a DSM represents a directed dependency. Consequently, the derived DSMs possess a matrix alignment symmetrical to the diagonal, i.e., if a dependency from person A to person B is established, the inverse dependency is created as well.

The textual extraction of the derived dependency meaning is explained in Figure 3-14 by coloring different parts of the network meaning according to the corresponding matrix content. The entire concluded network meaning results from the combination of these building blocks.

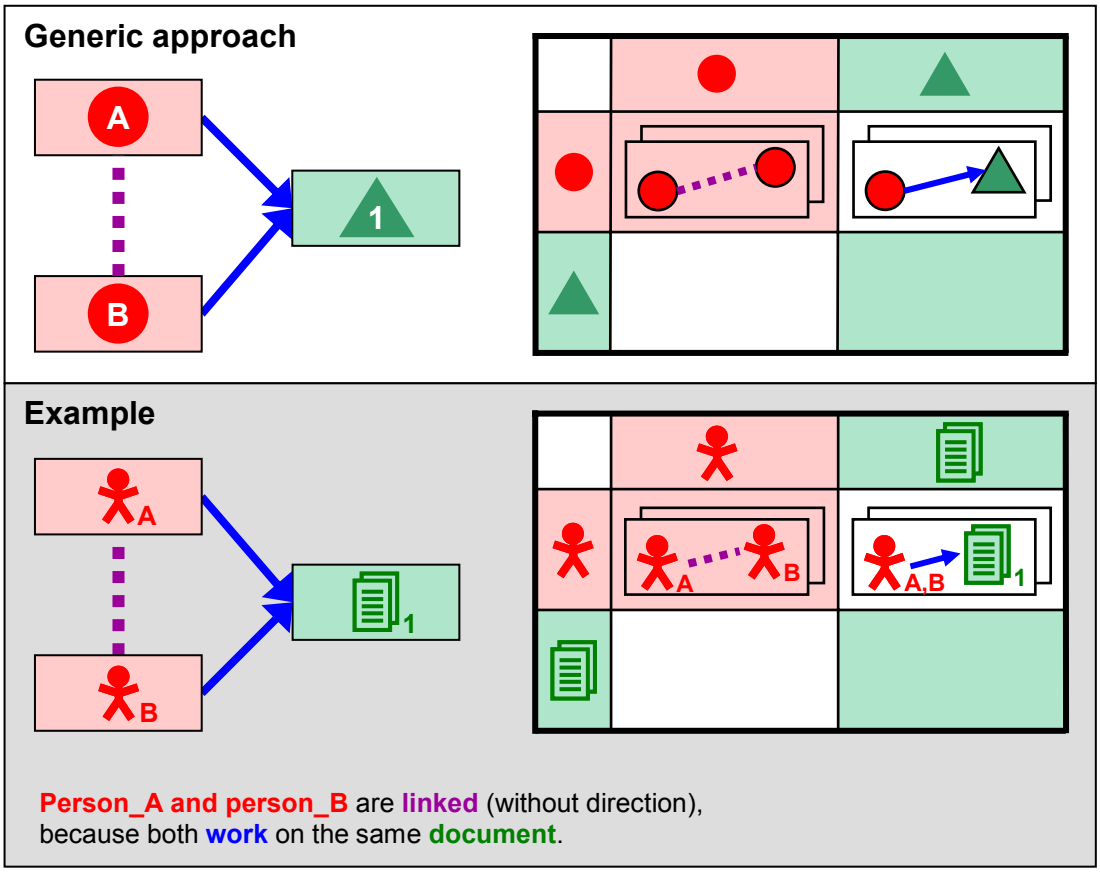


Figure 3-14 DSM derived from one DMM (Case 1)

Case 2 (Single DMM construction 2): This deduction of a DSM by one DMM (see Figure 3-15) corresponds to the procedure applied in Case 1. The only difference is the linking direction between both domains. If network impacts are interpreted in the matrix as "row element affects column element"[28], the influence from the second to the first domain (in the example chosen from documents to people) is located in the lower left DMM (instead of the upper right DMM in Case 1). As already mentioned for Case 1, the resulting DSM does not possess a linking direction, as the implied informational content only specifies the connection to the same document. The significance in the example's context shows a connection between two people, if they both require the same document. Of course, the derived dependency meaning is based on the dependency meaning of the applied DMM. For other use cases, the dependency meaning can be determined by using the textual building blocks in Figure 3-15 and replacing the phrases according to the individual situation.

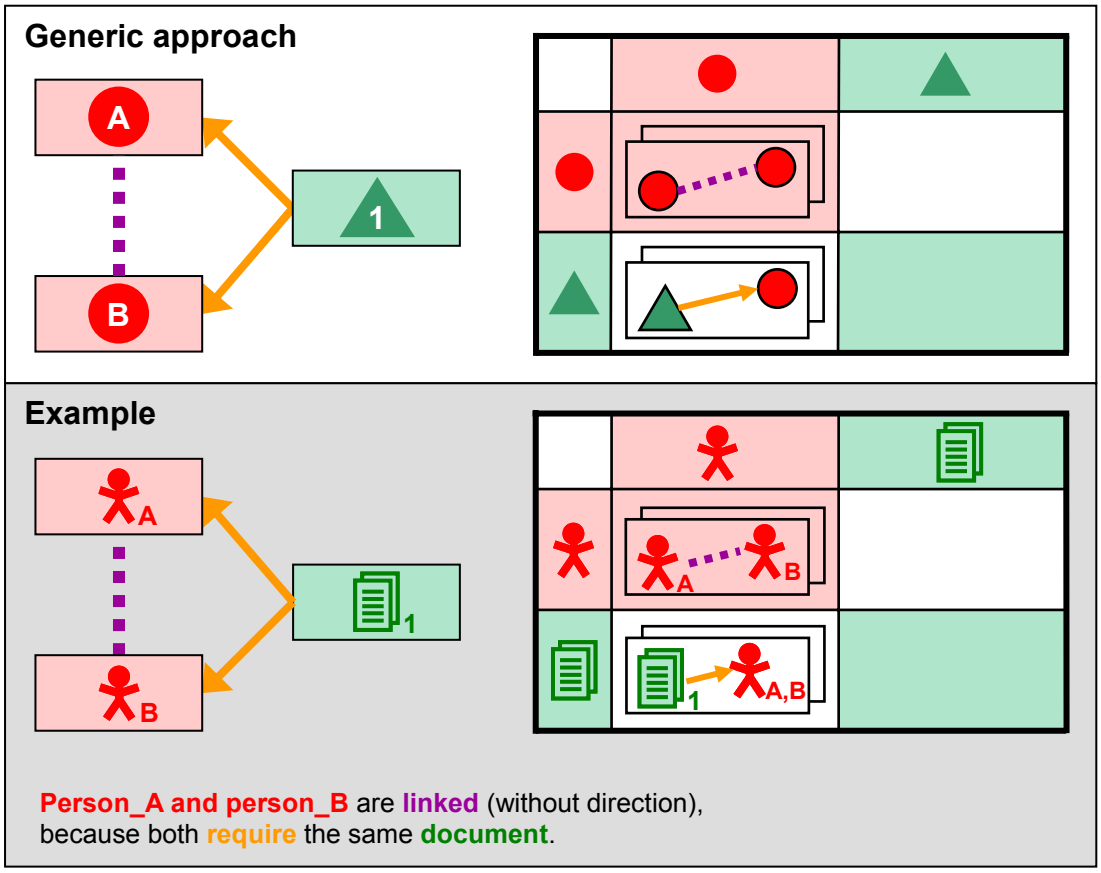


Figure 3-15 DSM derived from one DMM (Case 2)

[28] Some DSM approaches define the direction of dependencies in inverse order, i.e., "column affects row".

Case 3 (Dual DMM construction): A DSM can be derived from information stored in two opposed DMMs. While one DMM links the first domain to the second one, the other one links in the opposite direction. In contrast to the Cases 1 and 2, the derived DSM also provides information about the directions of dependencies, i.e., the resulting matrix is normally not symmetrical to the matrix diagonal. Figure 3-16 shows that a dependency directed from person A to person B can be determined, because person A works on document 1, which is required by person B. In other words, the indirect dependency between both people passing through document 1 is explicitly represented. The textual dependency meaning can be adapted to user-defined applications, if relevant building blocks are replaced by specific dependency and domain meanings.

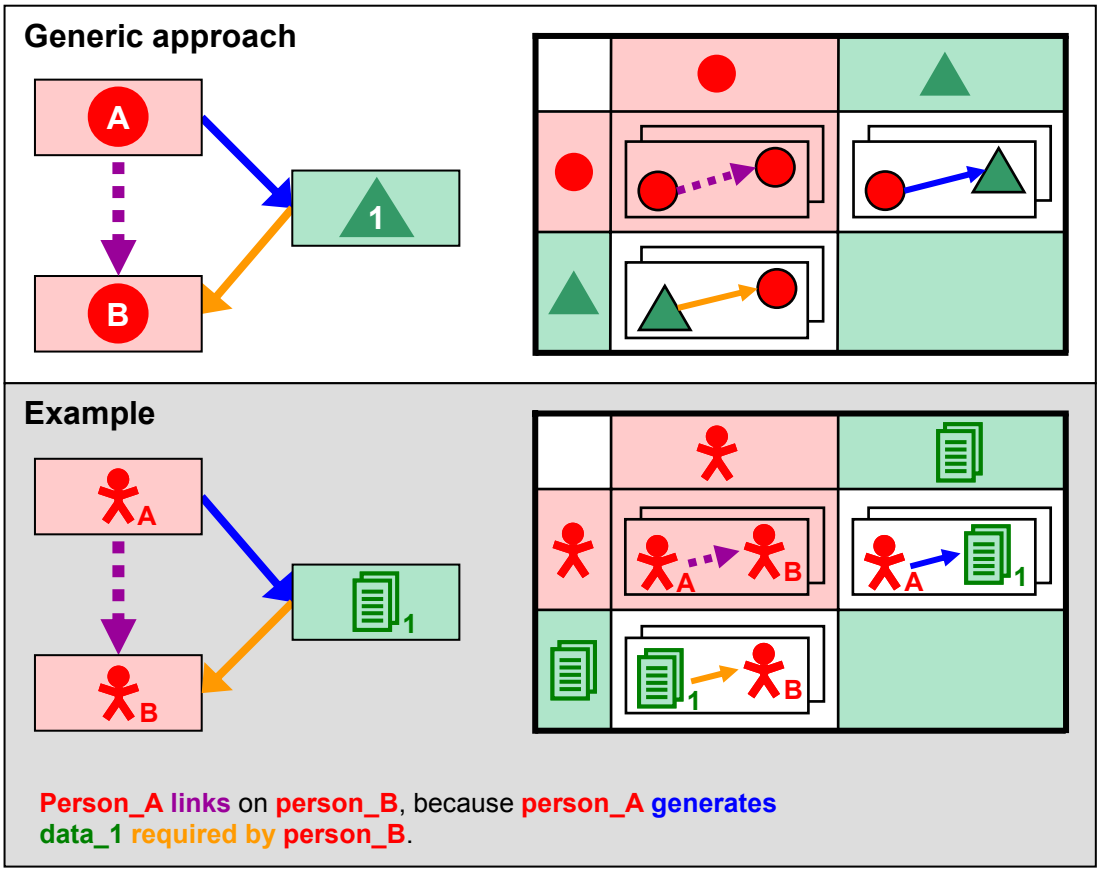


Figure 3-16 DSM derived from two DMMs (Case 3)

Case 4 (Combined DSM and DMM construction 1): A DSM can be derived from information stored in one DMM (connecting the first and the second domain) and one DSM (representing dependencies within the second domain). That means that resulting dependencies in the derived DSM (of the first domain) are condensed from relations between both domains as well as in between the second domain. In the example chosen in Figure 3-17, person A and person B both work on separate documents, as can be understood from the information of the DMM; person A works on document 1, which is required as input for document 2 (depicted in the DSM); this document is in the responsibility of person B. Consequently, a dependency between person A and person B can be derived, because adaptations executed by person A affect person B by the indirect linking of the documents.

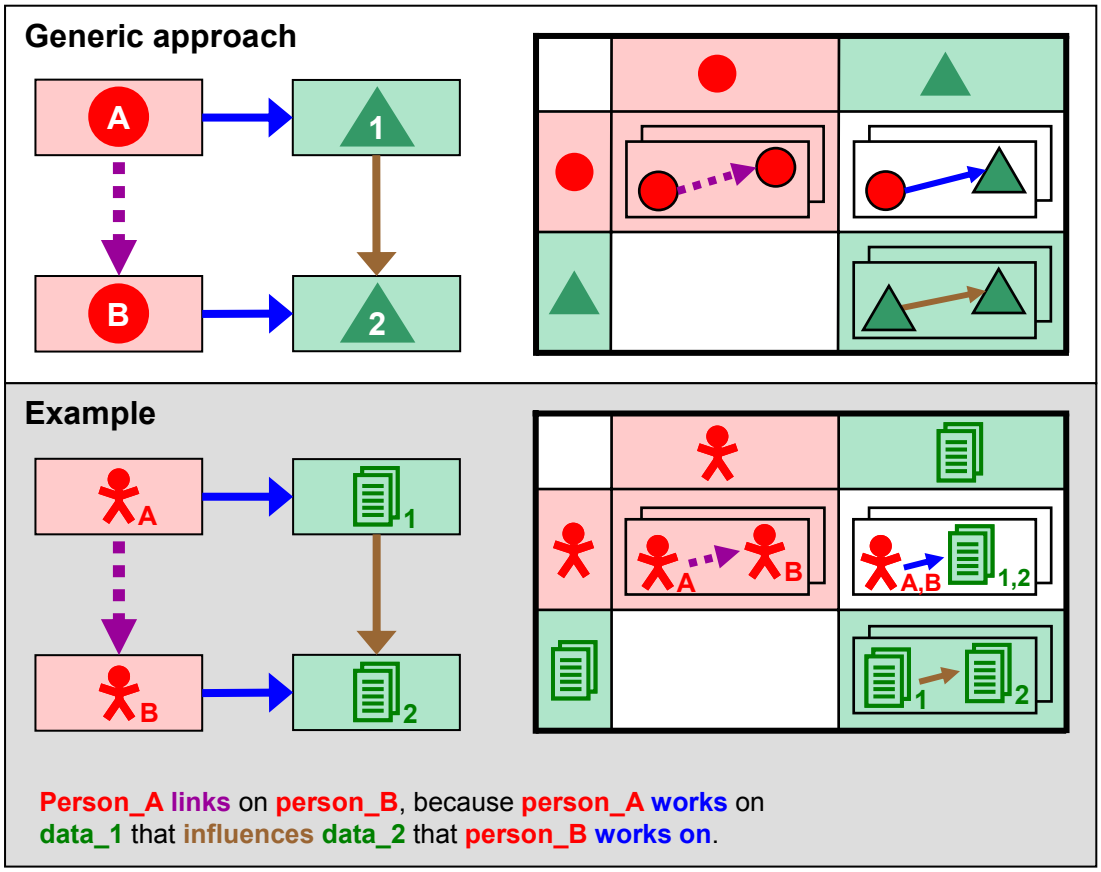


Figure 3-17 DSM derived from one DMM and one DSM (Case 4)

Case 5 (Combined DSM and DMM construction 2): This deduction of a DSM from one DMM and one DSM widely corresponds to the one represented by Case 4. As can be seen in Figure 3-18, the difference between both is the linking direction between the domains, expressed by the applied DMM. In the use case considered here, DMM information represents dependencies directed from the second to the first domain. In the example of Figure 3-18, person A and person B both require separate documents; however document 1, which is required by person A, provides input for document 2, which is required by person B. Consequently, a dependency between person A and person B can be derived, because changes executed to document 1 (required by person A) affect person B by the indirect linking of the documents.

It must be mentioned that the difference between Case 4 and Case 5 only results from the implied dependency meaning in the DMMs. Thus, the same information can be represented in Case 4 as in Case 5, if the dependency meaning "requires" is applied to the upper left DMM instead of "works on". However, for practical application it is appropriate to apply passively formulated dependency meanings to DMMs below the diagonal of a MDM, as this corresponds to the reading direction. This makes it easier for users to understand larger MDM representations with several domains and many filled sub matrices.

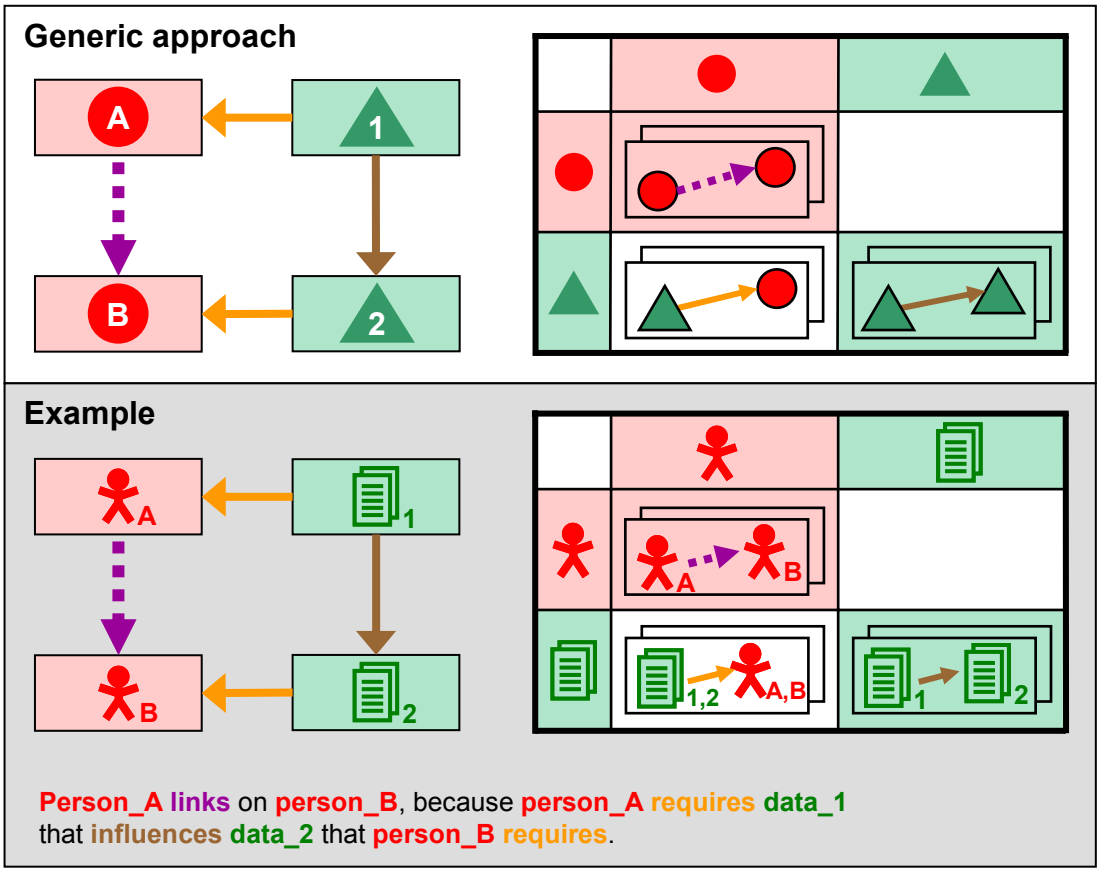


Figure 3-18 DSM derived from one DMM and one DSM (Case 5)

Case 6 (Combined DSM and dual DMM construction): A DSM can be derived from two DMMs (connecting the first and the second domain in both directions) and one DSM (representing dependencies within the second domain). That means that resulting dependencies in the DSM (of the first domain) are condensed from opposite oriented relations between both domains as well as within the second domain. This case combines three different dependency meanings into a complex one in the derived DSM. In the example of Figure 3-19, a deduced dependency is directed from person A to person B, because person A works on document 1, which is required for the compilation of document 2, and document 2 is required by person B. Consequently, a dependency directed from person A to person B can be derived, because the changes to the output of person A are propagated by documents and result in changing the input for person B.

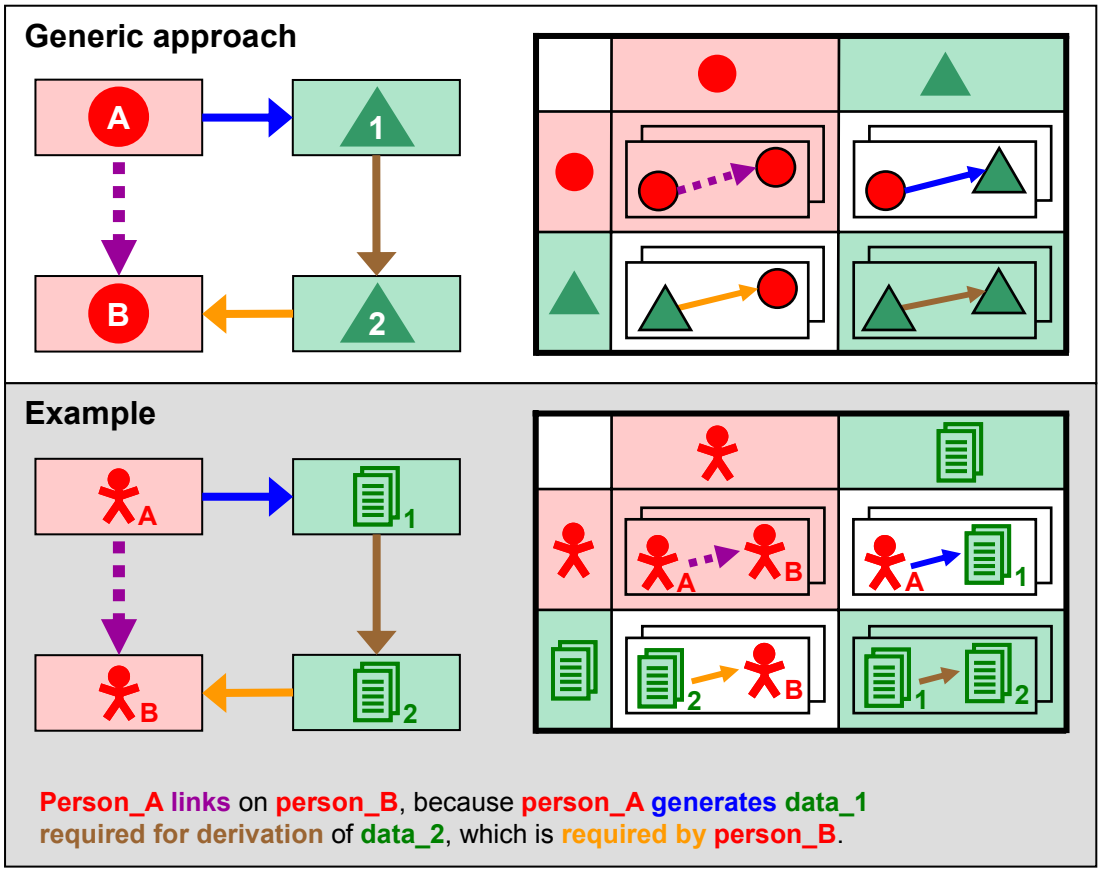


Figure 3-19 DSM derived from two DMMs and one DSM (Case 6)

The logic of DSM deduction from MDM modeling provides far-reaching possibilities for the consideration of complex systems. On the one hand, the general MDM layout affords a systematized depiction of comprehensive models. On the other hand, the deduction of DSMs offers a methodical access to specific system views.

Two fundamentally different approaches for practical application of the MDM can be identified, as shown in Figure 3-20.

The first one can be seen as an opportunistic application (right side of Figure 3-20) and can help to determine obtainable linking information from available data sets. This approach can also help answer the question: Which information on structural connectivity can be achieved by a combination of existing linkages? Available system information is classified in the subsets of a MDM, and possible deductions of DSMs are systematically computed (according to the available six cases of DSM determination). DSMs that are useful for further analyses can be selected by examination of the derived dependency meaning (see Section 3.5). This approach seems appropriate if information on the system structure is already on hand (e.g., by data imported from preceding system modeling approaches) and further information acquisition is out of question (e.g., because of unavailable resources or expert knowledge that can be captured by interviews). The systematic computation of all derivable DSMs from available MDM subsets can be efficiently realized by software support. However, it must be taken into account that the number of derivable DSMs increases disproportionately to the quantity of modeled domains and available subsets of a MDM. Even if the computation of all derivable DSMs can be executed, the selection of useful DSMs for further analysis and interpretation becomes time-consuming in larger MDMs. Section 3.5 provides a useful strategy, which permits the reduction of the number of DSMs that have to be derived and evaluated in practical applications.

The second approach of the MDM application is driven by case-specific requirements and depicted at the left side of Figure 3-20. The appropriate application scenario differs from the first approach in the non-availability of information about the system structure. If this is not on hand, then the specifically required system views (DSMs) must first be defined that help answer the case-specific questions. Based on this, subsets of the MDM can be determined, which are required for deriving the DSMs; typically several possibilities exist in such a case. The following question can be addressed by this approach: What information about system elements and their connectivity has to be acquired in order to derive a specific view of the structural dependencies? The procedure guarantees that only system information will be acquired that directly serves to determine the specific system view. This is important, as information acquisition can be extremely time-consuming and therefore must be executed in a target-oriented manner.

- First the required DSM has to be determined, and then its specifically required dependency logic has to be defined.
- Eventually a further domain must be considered, which allows deriving the requested dependency meaning by MDM computations. That means that the starting point of this approach is not a MDM, but a conventional DSM comprising the domain in question (e.g., people, because people's dependencies have to be analyzed).
- Depending on the required dependency logic, the DSM must be extended to a MDM by an additional domain (e.g., data, because people's dependencies by data transfer have to be analyzed).
- The MDM subsets must be determined in order to allow for the computation of the requested system view. Then, the textual dependency meanings of the six basic domain mapping logics can be applied. If the appropriate use case is identified, the

specifically adapted text blocks represent the dependency meanings required in the subsets for deriving the requested DSM.

As result of these steps, users are aware of the MDM subsets and the implied dependency meaning for the following acquisition workshops.

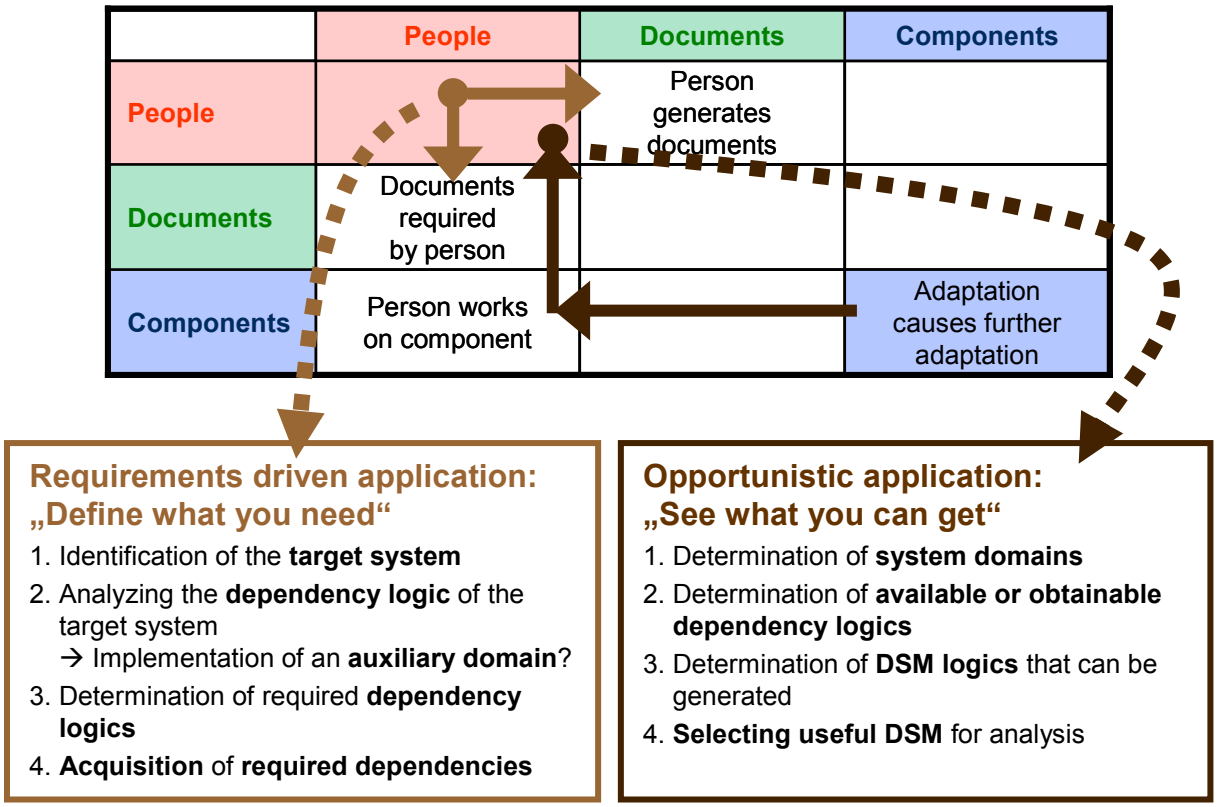


Figure 3-20 Application scenarios of the MDM

Both approaches of the MDM application are depicted in Figure 3-20: In case of an opportunistic application initially two subsets of a MDM are available. The change impact within components is known (representing a DSM), e.g., from parametric CAD-modeling. The assignment of people to components is known, and the dependency meaning is a “person works on a component”. This information could be taken, for example, from a PDM system. By applying both matrices to MDM computation, a people-based DSM can be derived without further information acquisition. The resulting network represents links between people who work on (different) components that are mutually dependent by change impact.

If this network does not help to answer a specific problem, the more time-consuming requirements-driven application of the MDM has to be applied. In the example chosen, the relevant question could, for example, request the people’s relationships according to their provision and request of documents. The question can be split up in two separate ones, both represented by individual DMMs in Figure 3-20: one DMM contains the documents generated by people, the other one, the documents required by people. After populating the DMMs with

data, the document-based people network (representing a DSM) can be derived by applying the domain mapping logic of case 3 (represented in Figure 3-16).

3.3 Information acquisition

The system definition, which was detailed in the previous section, provides the basis for the subsequent information acquisition. Generally, two different possibilities exist for capturing information about the connectivity within systems. If existing data sets can be used for an automatized acquisition of dependencies, less effort is required than in cases where dependency information is only available from the experience or implicit knowledge of developers. Typically, extraction of information that has not yet been documented requires time-consuming interviews in workshops.

Irrespective of the mode of information acquisition, the resulting data quality is of major importance to ensure that all further steps of the approach presented here on structure management are correct. As will be shown in a practical use case in Section 4.1.1, the existence or non-existence of two specific dependencies can significantly influence structural constellations and the behavior of an entire system. For this reason, requirements for assuring suitable data quality in acquisition processes will be considered next. Then details on the two different acquisition methods will be presented.

3.3.1 Requirements for assuring suitable data quality in acquisition processes

A high quality of captured data is the key factor for the accuracy and significance of all further structure interactions. Deficiencies in the acquisition of the data can not be corrected later on and often result in data that is useless for analysis and interpretation. In particular, if mistakes made during data acquisition remain undiscovered, they can become critical. In this case, even correctly executed network analyses can lead to erroneous findings that can cause misleading interpretations and unsuitable actions.

Often, interviews with experts have to be undertaken to extract implicit knowledge or experience knowledge. An important aspect of such information acquisition is the fact that the process is so time-consuming. Adequate workshop moderation and in-depth documenting of decisions made is required for effective execution. A moderator has to assure that workshop participants can concentrate on specifying the system's dependencies without being occupied with technical or organizational tasks. Workshop participants only need to see the two nodes whose dependency is actually discussed; such extraction of network parts asks for appropriate software support. The decisions made can then be transferred to a matrix or a graph in the background and can be applied in subsequent steps of the approach.

Besides organizing the general acquisition process, the moderator also has to assure a consistent understanding of the dependency logic during the whole process of information capturing. For example, a system comprising 100 elements requires 9 900 decisions about the existence of dependencies between two nodes. Processing these many decisions is time-consuming and typically requires several workshops. Therefore, it is understandable that the

effects of training and fatigue can influence the acquisition results. Without adequate support, participants' interpretation of the dependency logic may change (training effects) or some decisions may be based on insufficient considerations (effects of fatigue). Consequently, the workshop moderator has to remind participants of the dependency meaning discussed, for example, by formulating every question on possible dependencies.

The moderator needs to present system knowledge that permits questioning the reasons for the decisions made. This helps to improve the quality of acquisition, as workshop participants not only have to specify their decisions, but explain them, as well. In practice, two moderators are recommended for executing a workshop. One manages the organizational tasks and formulates the questions, the other records the dependencies as well as the explications. These manpower requirements are justified, as several workshop participants are required for an extensive period of time. That means that this kind of information acquisition is expensive, and therefore needs effective execution without any avoidable iteration.

Even if dependency information can be directly extracted from existing data sets without the need for personal interviews, a methodical procedure is required in order to guarantee adequate network quality. If data emerge from different sources, it has to be synchronized to avoid redundancy. Often, this represents a demanding task if different denotations exist for identical elements. Systematically implemented plausibility checks can help to verify the accuracy of imported data. However, this calls for the intervention of experts and limits the possibilities of an automatized data acquisition.

Data redundancy can also be helpful for improving the quality of acquired dependencies. If it is possible to compare redundant data sets emerging from different sources, equal information can help ensure its accuracy, and any contradictory information can be discussed with experts. This procedure helps to reduce time-consuming interviews to only those system dependencies that there is some doubt about.

Once system dependencies are combined in a network, some basic analysis criteria (see Section 3.5.3) make it possible to point out elements and dependencies whose structural embedding suggests their reconsideration. For example, a central element of a star-shaped structure represents an articulation node that can be identified by algorithmic support. Reconsideration of this specific constellation can result, for example, in the discovery that the articulation node depicts an element on a more abstract level than the other elements in the structure. For this reason the articulation node connects to a large quantity of other elements. This becomes clear in a practical example. If a structure comprises several electronic components and the general chassis of a car, almost all electronic components would be linked to the chassis by change impact. Changes to the specific components influence the superordinate, abstract chassis in most cases. In such situations, subdividing the abstract element can improve the modeled structure.

EICHINGER ET AL. describe a software-supported method to determine and represent indirect dependencies in acquisition matrices [EICHINGER ET AL. 2006]. This can be used for plausibility checks on existing structures. The authors base their proceeding on the hypothesis that in many use cases the existence of a direct dependency between two elements becomes more likely the more indirect dependencies exist between them. This assumption helps identify relevant element pairings for an iterative consideration.

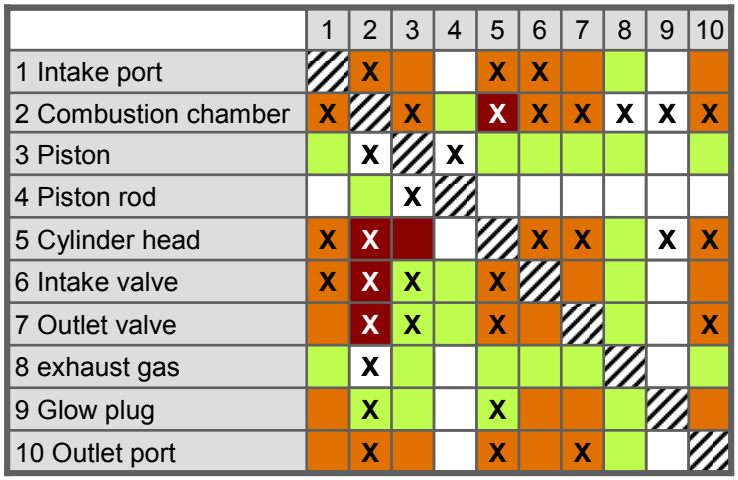


Figure 3-21 Visualization of indirect dependencies for plausibility checks

Figure 3-21 shows an example of the analysis of indirect dependencies. Matrix cells marked in (dark) red point out the element pairings possessing the highest amount of indirect dependencies, whereas orange (medium-colored) and green (light-colored) cells depict medium and low quantities of indirect dependencies. It can be seen that five matrix cells are marked in red; however, one does not possess a direct dependency and should therefore be reviewed in a plausibility check. In contrast, seven matrix cells possess direct dependencies, and no indirect dependencies are indicated. Consequently, the corresponding element pairings should also be reviewed concerning the accuracy of the decisions made. Plausibility checks comprise part of structure analyses and are based on mathematical possibilities of the graph theory. Detailed descriptions of the topic can be seen in Section 3.5.

Several requirements can be formulated for the effective process of information acquisition. Detailed descriptions of both general acquisition methods are presented in the following section.

3.3.2 Information acquisition by extraction from available data sets

Extracting information from existing data sets is a convenient method in those situations where information is already systematically described. Unfortunately, in most cases the required network will not be directly available, but must be derived from available MDM subsets and the application of the computational use cases presented in Section 3.2 (corresponding equations are shown in Section 3.5).

Many methods apply the connectivity between system elements, either by explicit matrices or at least by systematic lists (linking attributes to system elements). For example, the Quality Function Deployment (QFD) includes a matrix alignment connecting requirements, components, and elements of the production system with product attributes [AKAO 1992]. Thus, if the OFD is already applied in a development process, the information on connectivity can be directly imported from the QFD to subsets of a MDM. Impact matrices provide the

linkages within, for example, components; consequently, they represent DSM structures, which can be used to fill the DSM subsets of a MDM. Such impact matrices are often applied in comprehensive methodologies, such as the scenario technique [GAUSEMEIER ET AL. 1995]. In the TRIZ methodology, relation-oriented functional modeling is applied that can provide dependencies between functions ([ALTSCHULLER 1984], [TERNINKO ET AL. 1998, P. 47FF]). Besides this specific modeling, the application of flow-oriented and use-case functional modeling can also provide the connectivity in between components and functions respectively for users and use cases.

Additionally, many different ordering schemes can serve as sources for dependency information, if they are at least two dimensions. In these schemes, different characteristics get linked to each other. LINDEMANN describes ordering schemes as a method to combine the specifications of two different attributes and clarifies the procedure by the use of component variants [LINDEMANN 2007, P. 148F]. Such matrices can be seen as DMMs and imported in a MDM approach. Further information sources are available from the methodical applications of lists, such as requirement lists or FMEA forms. As the creation of such documents is often mandatory in development processes, they are generally available.

In product development processes, a lot of data is stored in several software tools, which can provide relevant input for filling MDM subsets. As tools are often applied as isolated applications, redundant data may occur. Software applications that can be used for extracting structural dependencies are often applied for project management, process management, or product data management. Furthermore, brainstorming tools and information on the parametric design can provide structure-related content.

Information acquisition by automated data import requires software support, as the amount of data which typically occurs can not be handled manually and specific interfaces must be implemented. Most software tools that are used for method application in product development already provide standard interfaces which can be exported to commonly used formats. Consequently, a large number of software tools can be applied with only a few interfaces.

3.3.3 Information acquisition from interviews

If suitable data on the structure in question is not directly available, information acquisition can only be realized by executing interviews with experts. In this case, required data often represents experience or implicit knowledge, whose structured acquisition can be problematic. Experts often know relevant parameters and dependencies only implicitly and have difficulty describing them. Furthermore, the distinction between direct dependencies and indirect ones is not always obvious. People often tend to declare direct dependencies, only because the relation seems to be important, and the fact of an intermediating part is often neglected.

It is important to apply a systematic procedure to information acquisition in order to guarantee the completeness and adequate quality of the resulting connectivity. If dependencies between system elements are, for example, captured by drawing connectors in a graphical representation, the illustration quickly becomes complex; it then becomes increasingly difficult for users to identify, if all possible dependencies have been considered. This is

pointed out in Figure 3-22 by comparing the dependency representation between nine elements in the graph and matrix. Both representations show the same content; however, it is difficult to identify element pairings that have not been considered so far in the graph. If the elements in the graph, in particular, do not align with an ordering scheme, elements can be difficult to find if networks become larger. Each element can possess dependencies to eight others, resulting in 72 possible dependencies for the entire structure. To be able to remember all possibilities, it is helpful to browse the matrix row by row (or alternatively column by column), as is suggested at the right side in Figure 3-22.

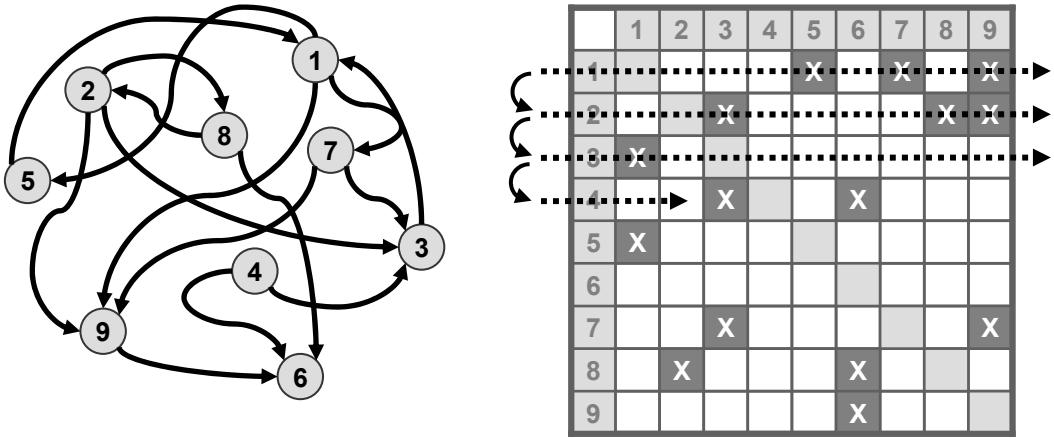


Figure 3-22 Suitability of graph and matrix for dependency acquisition

It is important to carefully plan the extent of systems in question: in order to elaborate a system's structure systematically, all possible dependencies must be surveyed concerning their possible existence. If, for example, a system consists of 30 elements, 870 element pairings must be discussed concerning probable dependencies between them. If it is assumed that one decision takes only 20 seconds, almost five hours are required to completely acquire the structure. This estimation does not even consider iteration steps or work breaks. Unfortunately, the quantity of possible dependencies increases geometrically with the quantity of involved system elements. For this reason, it is important to take only relevant elements into consideration. One limitation for the quantity of the elements considered (at the low end) is the extent of the depicted system. If a complex system is represented by a few elements only, obviously all elements will be mutually interconnected (e.g., in a model of an automobile composed by five building blocks). Of course, this might be a correct result; however, the system structure obtained does not provide further insight. Another constraint (at the top end) is the limit of practicability. As in the previous example, 30 system elements mean 870 decisions must be made. Consequently, a system comprising 100 elements requires 9 900 element pairings to be examined for possible dependencies. Obviously, this can not be completely processed during one single workshop. Assuming 20 seconds per decision, the acquisition would require more than 55 hours of work. Consequently, acquisition must be split up in several sessions. Besides the training and exhaustion effects already mentioned (see Section 3.3.1), workshop sessions that are presented at different times can transform people's understanding of dependency meaning during the acquisition process. Furthermore, the fact

that participants change from one workshop to another can not always be avoided. The workshop moderation and acquisition guidance must assure a consistent definition of elements and dependencies during the entire process.

In the following, a procedure of acquisition by interviews is proposed that takes into account the constraints previously mentioned.

The initial task is to collect relevant elements that form the system in question and therefore have to be considered in their mutual linking. A useful starting point could be to use the documents, such as the Bill of Materials or product catalogues. Elements must be identified and classified subsequently. This can be done by the use of hierarchical tree structures supported by several software applications on Mind Mapping ([LINDEMANN 2007, P. 277], [BUZAN & BUZAN 2002]). Based on this structured collection, brainstorming sessions with experts can be executed to complete the scope of the system. For reasons of documentation and appropriate visualization, it is recommended supporting the collection of elements by suitable software tools.

Next the collected system elements have to be aligned in a DSM-like matrix. Here, the elements are kept in their grouped order, whereas the categories of the hierarchical classification represent the domains that allow the formation of MDM subsets. The general layout of this MDM then allows the dependency meanings in every subset to be determined.

Often subsets of the MDM do not permit assigning relevant dependency meanings. These matrix areas can be excluded from further information acquisition. This helps to decrease the required acquisition efforts. For all other MDM subsets, dependency meanings have to be defined, which must be consequently applied to the detailed information acquisition in the next step. An example of a MDM layout can be seen in Figure 4-7 in the context of a case study.

It is recommended that matrices be filled out column by column or row by row, as all passive and active dependencies respectively of one element are considered sequentially. It must be further considered whether dependencies can be captured with different weightings. Of course, information about the system increases by application of weightings, and the acquisition effort additionally required appears to be manageable (every element pairing must be examined anyway). However, users must declare them for all resulting dependencies with identical accuracy. Measuring the quantity of dependencies in realistic scenarios will often be impossible. Variation in the accuracy of declared weightings decreases the resulting data quality and can hardly be corrected later on (without extensive rework). Consequently, the weighting of dependencies should only be applied to specific use cases, and the quantity of different weighting levels should be kept to a minimum (e.g., the three weighting levels, low, medium, and high, can be used).

It is recommended that dependencies for all MDM subsets be collected separately, as the applied dependency logic, as well as the type of elements, remains identical during the acquisition of one subset. An alternative filling order considers both directions of an element pairing simultaneously. For example, if the possible dependency from a function to a component is examined, the opposite direction reaching from the component to the function is considered as well. Even if both dependencies are located in different MDM subsets, this

procedure seems appropriate if both dependency meanings are complementary. Specific software support that visualizes both linkage directions simultaneously is required, as reverse matrix cells can be located far away from each other, if a matrix comprises many elements.

It was mentioned before that acquisition of dependency information by graph representations is not recommended due to insufficient system overview. Nevertheless, several software applications allow for the easy generation of small dependency networks. In fact, as long as the quantity of elements and dependencies considered remains small, the graphical network creation is generally intuitive for users and can, for example, be applied for capturing the most obvious dependencies, such as in a team-based process. These dependencies can then be transferred to a matrix for systematic completion by considering the dependencies that have so far gone unnoticed.

Generally, the process of specifying dependencies by matrix application can be executed by use of paper-based forms or spreadsheet software. However, specialized software tools can facilitate the process. For example, it can be helpful for users, if matrix cells can be marked as visited after examination, even if no dependency has been implemented. This can guarantee that the actual state of network creation is always obvious, even if several workshops are required for establishing the entire structure or several people contribute to the network creation. Furthermore, it must be possible to add comments to every element pairing in the matrix (i.e., if no dependency is also implemented between two elements). If experts debate about the existence of a dependency, the reasons that led to the decision implemented must be reproducible in later system examinations.

3.4 Modeling

In the last section, approaches for the acquisition of dependency information were presented. Once information about the interconnectivity of system elements is on hand, objectives and requirements of suitable representation forms will change. These objectives and requirements are discussed in the following section and appropriate modeling forms are presented. Here, modeling concerns the possibility of mediating complex structural information for the users and permitting interactions, case-specific extractions, and a better understanding of the system. System analyses and optimizations presented later (see Sections 3.5 and 3.6) also make use of the possibilities of modeling as a means of representing results for the user. A suitable concept is shown that meets the requirements of the approach presented here, and the limits, problems, and challenges of the concept are discussed.

3.4.1 Structure modeling: Objectives and requirements

In Chapter 2 fundamentals of representing complex structures were introduced. Based on this knowledge, specific objectives and requirements of the modeling in the approach presented are considered. This allows for evaluation and selection of modeling techniques, as discussed below .

Generally, modeling helps improve understanding of a complex system and implies that the representation is intuitive. For example, complex dependency networks can explicitly be

modeled by mathematical vector descriptions, but these do not allow the structure to be observed in total. Intuitive presentation of information about structural significance means a depiction according to a prior user's experiences and common sense. Thus, the required time for familiarization with the applied structure representation can be reduced to a minimum.

Additional requirements arise regarding the users of the intended structure models. People who want to obtain an overview and general comprehension of a complex system are often not experts about detailed system specifications, but are in charge of management tasks. Whereas matrix depictions are commonly used for representing technically-oriented content, management tools apply aggregated visualizations by use of, for example, diagrams or graphs. Therefore, the additional use of such representations can improve system understanding.

A suitable modeling approach can depict specific parts of the structure and allow browsing through selected subsets. This is required, for example, if complex systems must be analyzed on a detailed level; in this case, not all components and aspects of the system can be mediated simultaneously. Users need to selectively focus on structure subsets or to consider the system from a specific point of view. This means that the possibility of case-specific filtering of structural content is required.

Complexity often results from the far-reaching interconnectivity of components often lying beyond the scope of one single expert. In such situations it may be misleading if only elements of specific subsets of a structure are considered for deriving measures of structure management or optimization. If the actions taken are not based on all relevant structural aspects, the consequences that follow may affect system parts beyond the observed subset and may only become known if they cause problems. Thus, it has to be possible to aggregate relevant information due to user-defined aspects.

The modeling approach must allow browsing sequentially user-defined dependency chains. For example, users may be interested in the potential change propagation starting from one specific element [CLARKSON ET AL. 2001]. A convenient method is to initially observe the element's direct surroundings, i.e., all elements directly connected to the element in question. If in this group of elements some are relevant for change propagation effects, they can be part of the focus of subsequent observation. This procedure corresponds to a step-by-step navigation through the structure by following relevant paths. The same procedure needs to be applicable in the opposite direction as well. In this case, an element is identified that provides a non-desired change impact. Typically, this represents an output parameter of the system, such as a measured value that is out of the desired range. Thus, the objective is to identify the system element, which initially causes the impact chain resulting in the undesired value. This can be realized by sequentially tracing back relevant dependency paths starting from the known output element.

It can be assumed that for the approach presented here the main objective of modeling complex structures is to allow users to achieve an overview and understanding of the system. Useful system modeling enables users to conceive of the system outline and dominant characteristics. Further requirements are depicting filtered extracts or aggregated views of the structure. The browsing through the structure has to be possible, as well. Considering these requirements the modeling by matrices and graphs is discussed next.

3.4.2 The scope of matrices for structure modeling

Even if a matrix contains only a few elements, it is difficult for a user to observe the underlying structure. This is clarified in Figure 3-23; in the matrix on the left side the dependencies within a system comprising eleven elements are shown. Even with this minimal amount of elements, it seems impossible to discover any structural characteristics from the matrix. On the contrary, the dependencies seem to be almost equally distributed among all the elements. The right matrix in Figure 3-23 depicts exactly the same system as on the left side. The only difference is the adapted alignment of the elements in the matrix (without any changes to the dependencies). Only from this depiction can the underlying system structure be clearly seen. The structure consists of two complete clusters[29], where all included elements are mutually linked. In addition, four elements form a completely interlinked hierarchy[30], and element 11 represents the top element. The example shows that an appropriate matrix alignment is necessary to permit the identification of implied structural characteristics. Only if such an alignment can be found, subsets like clusters become obvious. Many algorithms exist that provide possibilities of matrix realignment [BROWNING 2001]. Many authors even provide step-by-step instructions that not only allow for computational but also manual realignment of matrices, e.g., KUSIAK [KUSIAK 1999, P. 29FF].

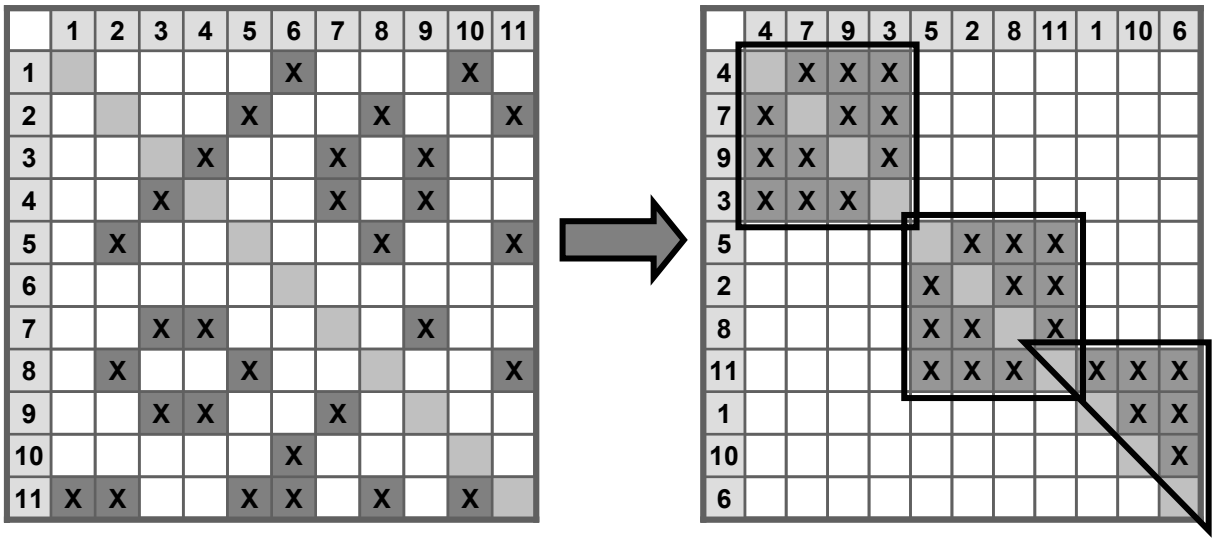


Figure 3-23 Discovering structural characteristics by appropriate matrix alignment

The detection of structural characteristics by matrix realignment is a powerful method that has proved to be useful in several practical applications [DANILOVIC & BROWNING 2004]. However, some disadvantages must be taken into account, which are explained in the following section.

[29] The structural constellation of a cluster is considered in Chapter 3.5.3 and in the Appendix.

[30] The structural constellation of a hierarchy is considered in Chapter 3.5.3 and in the Appendix.

Parallel depiction of structural features

Figure 3-24 shows an example of a small system structure of only seven elements. The initial alignment at the left side points out the existence of a cluster, including the elements 1 to 4. Further structural characteristics are not yet apparent from the depicted element alignment. However, if the elements in the matrix are realigned, a further structural characteristic can be seen. The right matrix in Figure 3-24 highlights a three-level hierarchy with element 1 as the top element (1 linking on 4 and 7, 4 linking on 5 and 2, 7 linking on 6 and 3). The example shows that different matrix alignments can highlight different structural content – and one alignment can prevent the detection of specific structural constellations. If the same elements are part of different subsets, it may be impossible to visually arrange them all at the same time. In fact, it is impossible to depict the cluster and the hierarchy of Figure 3-24 simultaneously in the matrix.

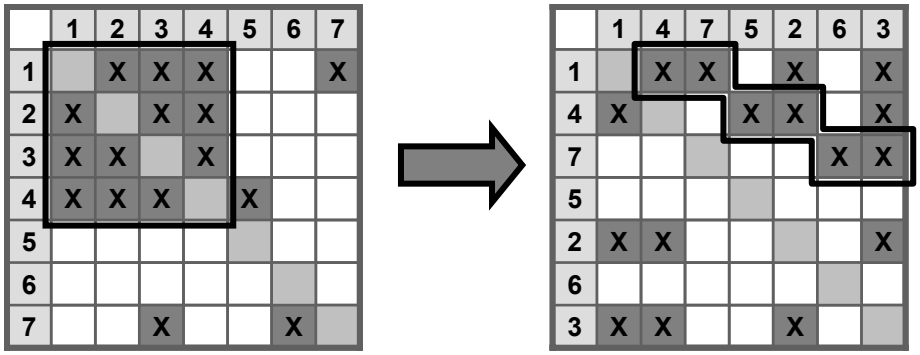


Figure 3-24 Representation of overlapping subsets

Of course, the matrix alignment does not affect the structure itself. That means that if an actual alignment hides a specific subset, it does still exist. However, the identification of one subset by matrix alignment can be misleading, as no information about the existence of further ones is available. For this reason, matrices are often improper representation forms for systems with a multitude of overlapping[31] structural characteristics.

Recognition of specific structural attributes

A similar example that visualizes the limits of matrix representations is shown in Figure 3-25. The system depicted is composed of nine elements; the matrix at the left side shows two complete clusters containing four elements (indicated as A and B) and a third one containing three elements (indicated as C_1). The clusters A and B as well as B and C_1 mutually overlap by one element (4 and 7). However, the system's structure overextends the matrix's capabilities of representation. This can be seen from the graph depiction at the right side of Figure 3-25, visualizing the same structure exactly. The graph clarifies that the system includes three clusters, each containing four elements and overlapping with the two other ones in one element. This specific constellation results in a fourth cluster comprising three

[31] The term "overlapping" refers to the occurrence of identical system elements in different structural subsets.

elements (1, 4, 7) and overlapping with all other clusters in two elements. This constellation can not be displayed intuitively in a matrix form, as explained by SHARMAN & YASSINE [SHARMAN & YASSINE 2004]. Only one cluster (here B) can be aligned in connection to two others (here A and C_1). As the cluster in the lower right corner of the matrix has to be linked to the clusters A and B, it becomes visually split up (indicated by C_1, C_2 and C_3 in Figure 3-25). The attempt to align all four elements (1, 7, 8, 9) belonging to cluster C side by side would split up cluster A in the matrix depiction. This shows that structures comprising highly interrelated subsets may require more possibilities for their representation than available in common matrices.

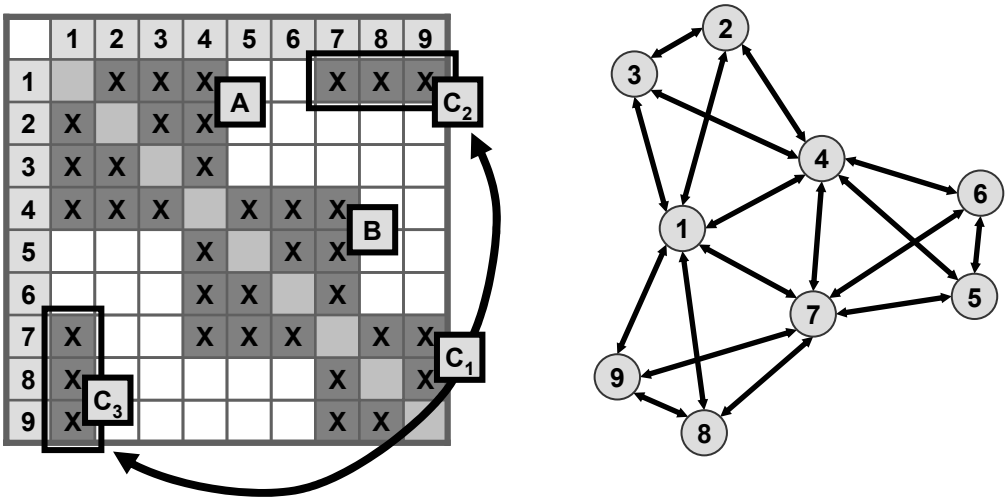


Figure 3-25 Limits of structure representation in matrix form (adapted from [SHARMAN & YASSINE 2004])

Another example of the deficiencies of matrix depictions is shown in Figure 3-26 (matrix) and Figure 3-27 (graph). The structure depicted could, for example, represent the communication flows between people in a development process. That means that the eight nodes of the structure represent the developers, and the dependencies between them indicate the document exchange between them. The practical use case might be that the management observed a stagnant flow of the development process, because most developers state they often wait for relevant information. By means of structure analysis, impeding constellations can be identified and possible optimization measures can be proposed. At the right side of Figure 3-26 the matrix elements have been realigned by the partitioning method[32], which results in one possible matrix alignment that possesses one dependency below the matrix diagonal. This indicates the existence of a feedback loop[33]. This could explain the stagnant information flow, as one developer requires information input that can only result from his own (preceding) information output to others.

[32] The partitioning method and its structural meaning are considered in Chapter 3.5.3 and in the Appendix.

[33] The structural constellation of a feedback loop is considered in Chapter 3.5.3 and in the Appendix.

However, the entire picture of the structure only becomes clear when it is visualized by a graph. From the left side of Figure 3-27 it can be seen that in total three feedback loops exist in the structure. Furthermore, they all overlap in two dependencies (developer 4 to 5 and developer 5 to 3). Generally, it is also possible to mediate this information by the matrix; however, it can not be clearly depicted. As mentioned before, the matrix alignment at the right side of Figure 3-26 represents only one possibility. In fact, a matrix alignment would also be possible where only the dependency directed from element 4 to 5 would be located below the diagonal (due to the existence of feedback loops in the structure, it is not possible to align all dependencies at one side of the diagonal). Only the knowledge about both possible alignments permits drawing the same conclusions from the matrix as the graph.

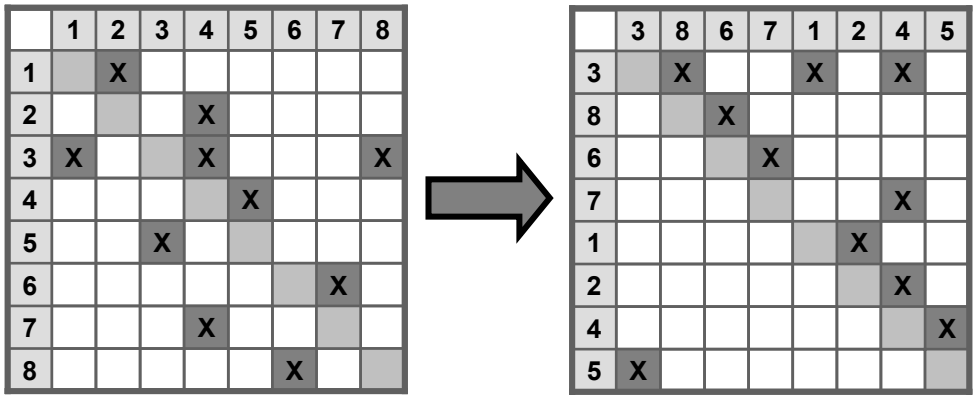


Figure 3-26 Matrix realignment for identifying feedback loops

The structural information from the graph depiction can determine a suitable optimization measure. Obviously, all three feedback loops can be eliminated by avoiding one of the two dependencies, where all three feedback loops overlap each other. The result obtained by eliminating the dependency between element 5 and 3 is depicted at the right side of Figure 3-27. Transferred to the practical use case, this structural change could be realized by enforcing the mutual communication between developers 5 and 3, for example, by seating them together in the same office. This would reduce the need for implementing an explicit information transfer between them, as short iteration steps in communication will guarantee their informational synchronization. In the optimized structure, developer 3 represents the person who initiates the process by starting three parallel information flows. The resulting information is collected by developer 4 and forwarded to developer 5.

Exactly the same optimization measure would result from the matrix depiction, because the dependency between developer 5 and 3 represents the exposed dependency located below the diagonal. However, the disadvantage of the matrix is that the second alternative for feedback loop elimination can not be seen in the same matrix alignment (dependency directed from developer 4 to 5), and therefore is probably neglected when considering possible optimization measures. Consequently, focusing on one specific dependency results from the randomly appearing matrix alignment.

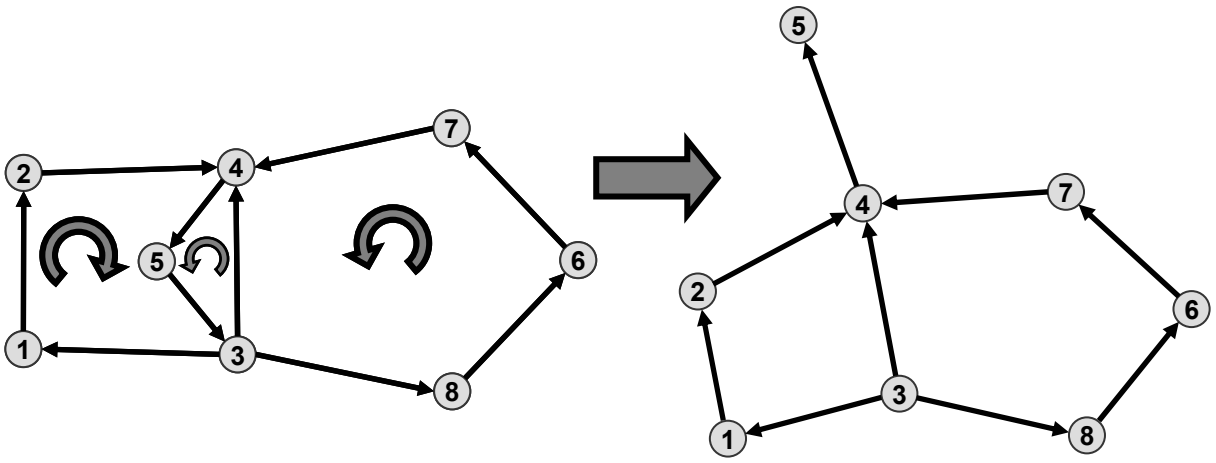


Figure 3-27 Overlapping feedback loops and possible optimization measure

Analysis of the overall system structure

A final example comparing matrix and graph representations is shown in Figure 3-28, which depicts the component structure of a ball pen. Here, a strength-based graph (see Section 2.2.2) is applied with non-directional dependencies. These are represented in the matrix as bi-directional dependencies, symmetrically aligned to the matrix diagonal (e.g., the "Tube" links to the "Distance bush" and the "Distance bush" links to the "Tube").

	1	2	3	4	5	6	7	8
1 Press button				X			X	
2 Spring				X				
3 Grip							X	
4 Ink store	X	X			X		X	
5 Nib guide				X			X	
6 Clip							X	
7 Tube	X		X	X	X	X		X
8 Distance bush							X	

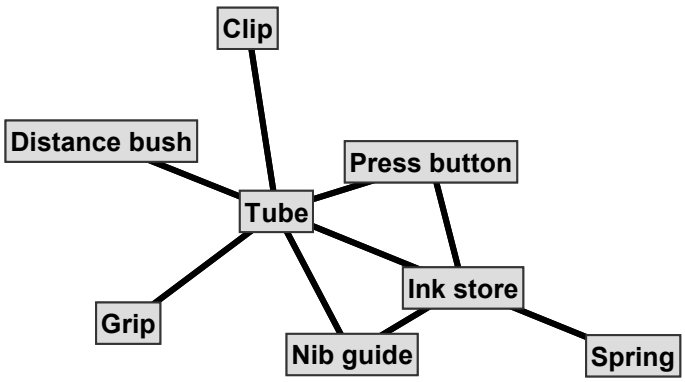


Figure 3-28 The structure of a ball pen represented in a matrix and a strength-based graph

Even if both representations contain the same information, the implied structure is easier to understand by the strength-based graph – the "Tube" represents the core element, as almost all other elements are linked to this. People can also extract this information directly from the matrix, if they are used to this depiction – the row and column associated with the "Tube" are the ones most filled with dependencies. In the graph representation, the "Tube" is located in the center, what makes its structural relevance intuitive. Two completely interlinked clusters exist in the structure ("Tube"–"Nib guide"–"Ink store", "Tube"–"Press button"–"Ink store")

that can not simultaneously be pointed out in the matrix; the appropriate element alignment for one cluster disrupts the alignment of the second one. In the graph, both clusters and the fact that they both overlap in the two elements “Tube” and “Ink store” can be seen.

Generally, the more elements that exist in a structure and the more interlinked these elements become, the less appropriate the matrix depictions seem to be. Overlapping of substructures becomes more likely the more relevant constellations exist in a structure. The successful identification of specific constellations by matrix alignment does not indicate the existence (or non-existence) of other ones. In fact, the depiction of one constellation can hide other ones.

The preceding examples suggest that (strength-based) graphs outmatch matrix depictions and qualify for the mediation of system structures for the user. Whereas matrices are suitable for purposes of information acquisition (see Section 3.3), in structure modeling they only seem to possess advantages for specific representations, such as isolated clusters. However, strength-based graphs also possess deficiencies that are presented in the following section. These deficiencies must be known in order to avoid misinterpretation of structure representations.

3.4.3 The scope of graphs for structure modeling

Figure 3-29 shows a strength-based graph (see Section 2.2.2) containing different edge weights. These weights create different forces of attraction between the elements and result in different edge lengths. As can be seen in the figure, the edge weight between elements 1 and 2 is ten times lower than the other two edge weights. It could be assumed that the edge length should also differ by the factor ten in order to present the structural content correctly. However, in the constellation presented, the lengths of both higher weighted edges will each be at least more than 50% of the lower weighted one for geometric reasons. This leads to the conclusion that the absolute length of edges in a strength-based graph can not be taken into account for analyses and interpretation of the structure.

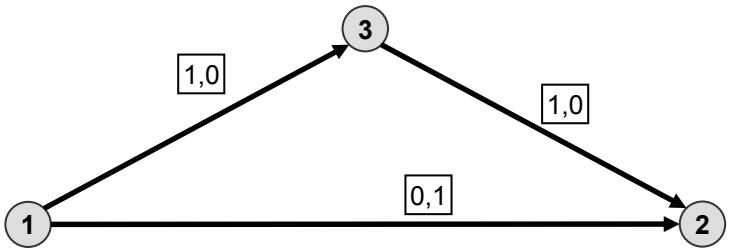


Figure 3-29 Correlation of edge weights and edge lengths

A similar use case is depicted in Figure 3-30, where the dependency meaning is slightly different than in the example before. Here, the weight of edges is interpreted as closeness between elements; the smaller the numbers, the closer two elements of the structure are related to each other. As was seen in Figure 3-29, a direct connection between two elements must always be shorter than an indirect one (a straight line is the shortest connection between two points). This results in an improper modeling in Figure 3-30, because the indirect linking

of element 1 to element 2 (via element 3) possesses a lower overall edge weight than the direct one (8 compared to 10).

The automatic alignment of the elements in strength-based graphs results from the mutual repulsive forces between all elements and the force of attraction due to dependencies that connect elements. In most cases, these system constraints do not necessarily result in one distinct alignment, but rather a multitude of arrangements that correspond to the existing forces. This effect can be seen in Figure 3-30, where element 3 is located above the dependency connecting the elements 1 and 2 (and closer to element 1 due to a stronger dependency than to element 2). Element 3 could be located below the dependency connecting the elements 1 and 2 as well, without causing any changes to the structural information.

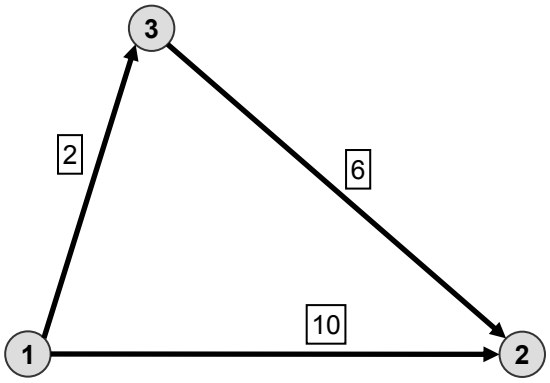


Figure 3-30 Correlation of edge weights and edge lengths – identification of the shortest path

While in Figure 3-30 the alignment of element 3 will not result in any misinterpretation, Figure 3-31 shows a structure where the interpretation of element positioning becomes more relevant. The elements 1 to 7 all possess dependencies directed to all elements of the structure, as can be seen in the matrix at the right side of the figure. In contrast, the elements 8 to 15 do not link to any elements of the structure. The resulting alignment in a strength-based graph is highly significant, even if single dependencies can hardly be distinguished—the graph is point symmetrical and clearly indicates two groups of elements. The first one includes the elements 1 to 7 (in the center of the graph) that are all mutually linked (also indicated by the framed and completely filled area in the matrix). The second group forms a circle around the inner group and contains all other elements of the system, representing linking targets for the elements of the first group.

The alignment in the graph allows for the intuitive comprehension of the two existing groups. However, further interpretations of the elements' positions within the groups would not be correct. For example, the positioning of element 7 in the middle of the inner group results randomly. Every element of the inner group could be located at this place instead, surrounded by the six other ones. Furthermore, the proximity between two specific elements in the graph does not possess any expressiveness; all elements can mutually change their places within the same group. For example, the elements 1 and 6 are located opposite to one another and possess the farthest possible distance from each other in the inner group of elements. However, the elements 6 and 3 could change their positions without changing the structural

content, resulting in a side-by-side alignment of elements 6 and 1. In addition, both element groups can be freely rotated around the center point independently from each other; this neither changes structural information nor violates the rules of the graph representation. That also means that the proximity of two elements from different groups does not provide any expressiveness of structural content. Even if the example in Figure 3-31 seems to be artificially constructed, a similar one will be considered in the case studies in Chapter 4 that emerged from a realistic development use case. The example indicated one problem when modeling structures with strength-based graphs. Even if the visualization provides an intuitive comprehension of structural content, users must be aware of the limits of interpretation. In this context, ideas about the future enhancement of strength-based graphs will be addressed in Chapter 5.

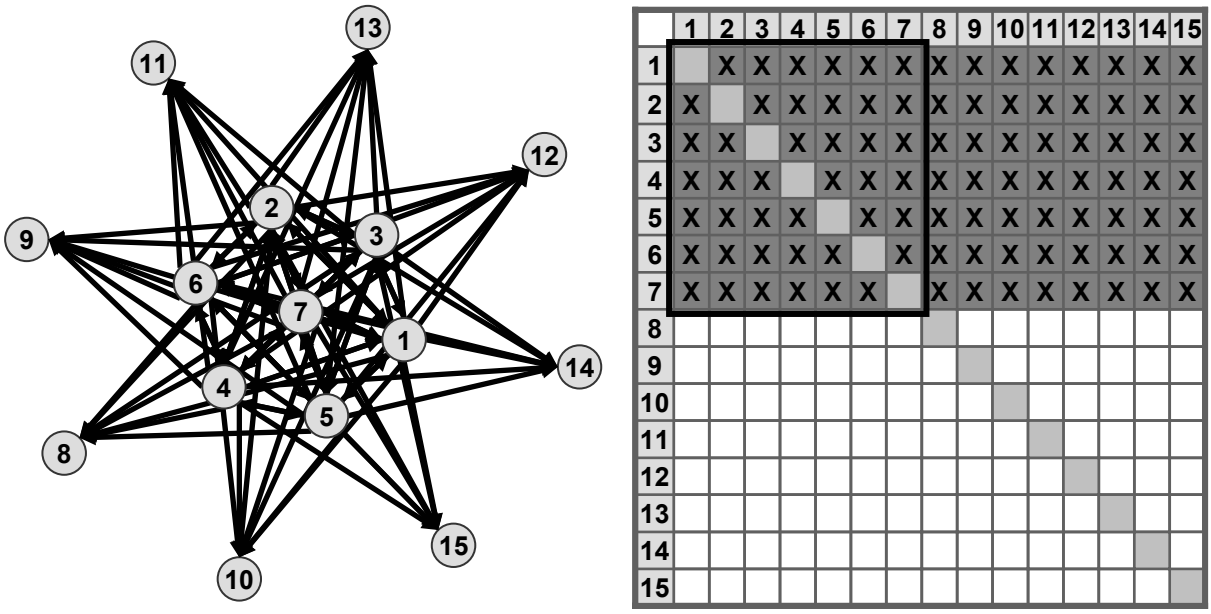


Figure 3-31 Significance of element alignment in strength-based graphs

3.4.4 Selection of an appropriate modeling technique

After describing available modeling techniques, their suitability for the application scenario, and their advantages and deficiencies, a modeling approach must be determined for the approach presented. As already described in Sections 3.2 and 3.3, matrix-based techniques are indispensable; the system definition of MDM is founded on matrices and can not be easily replaced by other modeling forms. Additionally, information acquisition requires a systematic system depiction that allows the sequential examination of all possible dependencies. The application of matrices is well established and well suited for this purpose. As matrices are already implemented in a multitude of product development methods, people are used to the representation form. In many cases, it is possible to directly transfer dependency information from other matrix-based methods to information acquisition of system modeling (which reduces acquisition efforts). Matrix-based analysis methods have existed for many years and

permit efficient analyses of system structures. The fundamentals of the DSM are particularly important for the system analysis of the approach presented here (see Section 3.5). The application of DMM approaches has increased lately and is especially helpful for efficient information acquisition; however, only a few analysis algorithms can be applied to such matrices so far.

Matrix representations of system structures possess deficiencies that have been clarified in this section. Because of these deficiencies, the approach presented here towards structure management asks for a supplementary modeling by graphs, in particular strength-based graphs. These fulfill the requirements for the intuitive comprehension of visualized structures, enhance the capabilities of matrices concerning the mediation of structural subsets, and allow for extensive possibilities of structure interaction. Users can easily select specific subsets for closer consideration or navigate through the structure step-by-step. Thus, strength-based graphs are also appropriate for users without a technical background and can help enhance team work on complex systems. Finally, graphs and matrices can easily be transformed into each other. For this reason they represent the best combination for improved modeling of complex systems and are applied for the structure management approach presented here.

3.5 Structure analysis and evaluation

In the preceding sections fundamental requirements for the analysis of complex system structures were discussed. Starting with the appropriate system definition, promising procedures for information acquisition were presented. Subsequently an approach for modeling complex structures was derived from available techniques. Once structural information is on hand, an effective analysis approach serves to identify structural attributes. These comprise part of almost all systems and form comprehensive system behavior or characteristics, which are not assigned directly to specific components, e.g., the system reliability. In addition to the reliability of the single components, the reliability of the entire system can vary depending on the system's structure. For example, for (standard) Christmas lights, the system's overall reliability is typically rather low, because of the sequential alignment of the bulbs. If one single bulb fails, the function of the entire system fails too. A change of the product structure for parallel wiring would dramatically increase the system's reliability, using the same bulbs (unfortunately, product costs would increase as well). Even if this example seems to be rather simple, similar structural characteristics can often be found in more complicated systems. In historically grown systems in particular that have been adapted several times, structures often do not serve the actual needs and possess potential for optimization. In these cases, system components have been updated or changed in the past without any adaptations of the basic structure, because this would often have required major rework. This results in up-to-date technical components in an out-of-date constellation. Such situations can be found in product design as well as in organizational or process structures. The consequences are structures that do not suit the actual requirements of the system.

As structure analysis serves for the identification of structural attributes, it helps characterize specific system elements, their embedding in the structure, and the entire system. This allows for a comparison of similar functional products by referring to their structural design. Products with different functionality and with another scope can also be compared using

structural attributes in terms of a benchmark. The structure analysis itself does not provide any direct benefit, but represents the fundamentals for structure interpretation, better structure management, and practical, target-oriented structure improvement. A structure manual can be created based on the analysis results, which can support users by enhancing their communication during product development. The structure manual can also help users become aware of dependencies, which are likely to remain unconsidered or unknown to developers. Furthermore, both opportunities and restrictions for system adaptations can be derived from a structure manual. Thus, developers can evaluate whether specifically planned adaptations are recommended for execution or whether they would result in unmanageable change propagation. In addition, structural areas can be identified which are appropriate for adaptations, because the risk of change propagation is low.

Besides creating the structure manual, the structure analysis is the basis for the identification of structural potentials. Analyses can point out parts, whose adaptation can lead to significant system improvements. This means an adjustment of the structure from the desired system requirements. Appropriate algorithms can be applied to the structure in order to obtain better system reliability or robustness.

Figure 3-32 shows the integration of structure analysis in the overall process of managing complex structures (see Figure 3-1). The input for the structure analysis is formatted network data comprising elements and their dependencies. The approach presented here only focuses on intra-domain analyses (see Section 3.2), as most analysis algorithms require this restriction (an exception would be, for example, the DMM clustering [DANILOVIC & BROWNING 2004]). The integrated consideration of multiple domains in analyses would also make it difficult for users to derive interpretations. However, multiple domains can generally be decomposed to intra-domain use cases described by the system definition in Section 3.2.

As can be seen in Figure 3-32, basic analysis methods are applied to the network data received through information acquisition. The output of these methods is knowledge about structural characteristics of the system in question. Subsequently methods for the discussion of practices can be applied in order to obtain a structure manual for improved system management or structure potentials to enable developers to improve the system structure (see Section 3.6).

The analysis of DSM structures can be classified in three steps, which have to be applied sequentially: provision of DSM structures, specification of analysis objective, and method application of DSM structures for the identification of structural characteristics. At first, a system structure comprised of one domain and one dependency type must be prepared. If such a DSM-conforming structure has been acquired directly through interview or by extraction from other data sources, this process step is dispensable. In all other cases, it must be derived according to the general logic of deriving DSM by MDM explained in Section 3.2. According to the use cases presented in Section 3.2, six computational schemes exist and are presented in the following section. The figures relate directly to the corresponding ones in Section 3.2. The equations required for computation are based on the fundamental matrix multiplication.

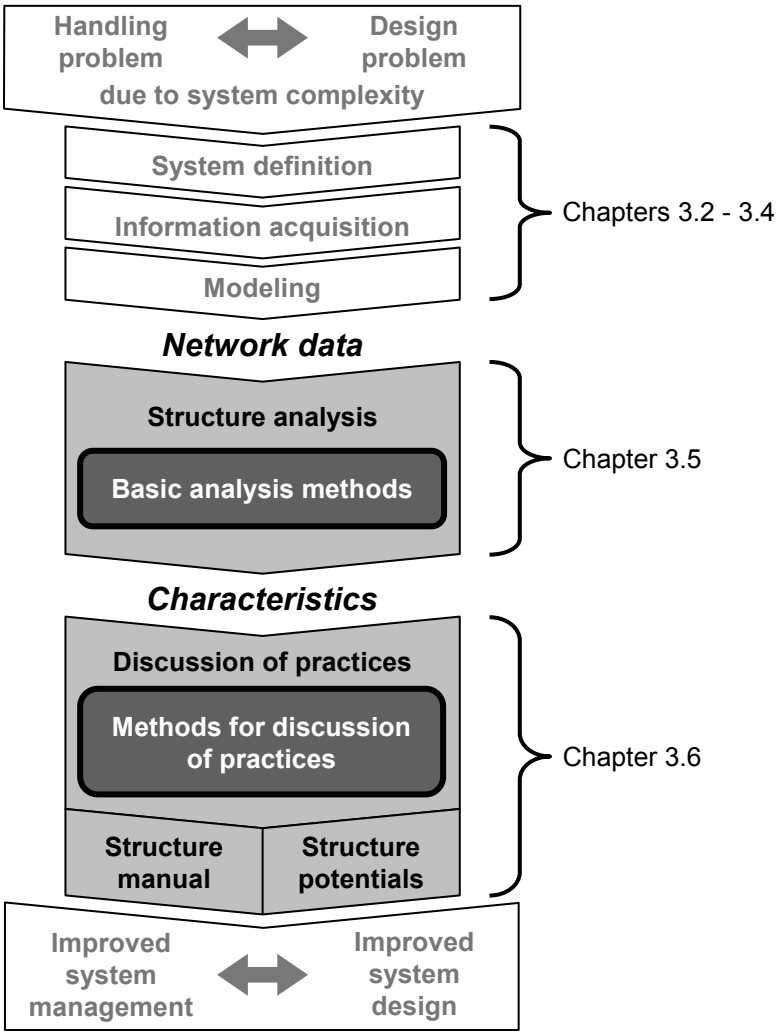


Figure 3-32 Integrating methods of structure analysis to the general structure management approach

3.5.1 Computation of DSMs from MDM subsets

Figure 3-33 to Figure 3-38 show the logic dependencies at the left side and the corresponding matrix depiction at the right side. The letters in the sub-matrices are used to explain the equations mentioned below each figure. X and Y indicate DSMs that represent the computation objectives. One of these DSMs can be part of the input data for deriving the second DSM (Cases 4, 5, and 6 in Figure 3-36 to Figure 3-38). G and H indicate DMMs filled by information acquisition, which serve as input data. G^T and H^T describe the transformed matrices of G and H, which are required in cases 1, 2, 4, and 5 for computational reasons. These transformed matrices do not contain any new information concerning the original matrices G and H. Generally, transformed matrices can be directly derived from any matrix (DSM or DMM). The logic computation of the desired DSMs is shown by using a MDM comprising two domains only. Even if MDMs will occur in practice which contain more than

two domains, all possible deductions of the DSMs can be executed by combinations of the computations presented.

In all use cases the schemes for the computation of both DSMs X and Y are mentioned, even if they only differ in the order of matrix multiplication. It is also possible to switch the alignment of domains in the MDM and then apply only one computation scheme for every case presented. That means that in case the lower right DSM Y has to be computed, the alignment of both MDM domains must be switched first; the DSM requested is now located at the upper left side and the computational scheme for the DSM X can be applied. In the following, only the computational schemes and assigned equations are presented. Examples that clarify the significance of resulting networks and the dependency meaning can be seen in the corresponding use cases shown in Section 3.2.

Case 1 (Single DMM construction 1): The easiest use case represents the determination of an intra-domain matrix from one inter-domain matrix. Figure 3-33 depicts the domain mapping logic at the left side and the matrices involved at the right side. Matrix G must be filled with data by information acquisition; equation 3-1 shows the required computation for the objective DSM X. The transformed matrix G^T is required and can be computed directly from matrix G without the need for further information. If matrix Y represents the objective matrix, equation 3-2 shows the required computation that differs from equation 3-1 in the inverted order of matrix multiplication. The resulting network does not provide any linking direction; that means that the resulting matrix X only possesses dependencies symmetrically aligned to the diagonal.

The application of this computation will normally not provide extraordinary findings, and common filters implemented from data bases or spreadsheet software provide similar possibilities. However, the visualization of achievable networks can be useful in graph or matrix depiction for highlighting groups of intensely related elements. As the resulting network does not contain linking directions, most analysis algorithms (presented later in this chapter) can not be applied or do not lead to further system insight. For example, elements that are mutually linked always form complete clusters, because no linking direction is specified. Therefore, cluster analyses will not be helpful.

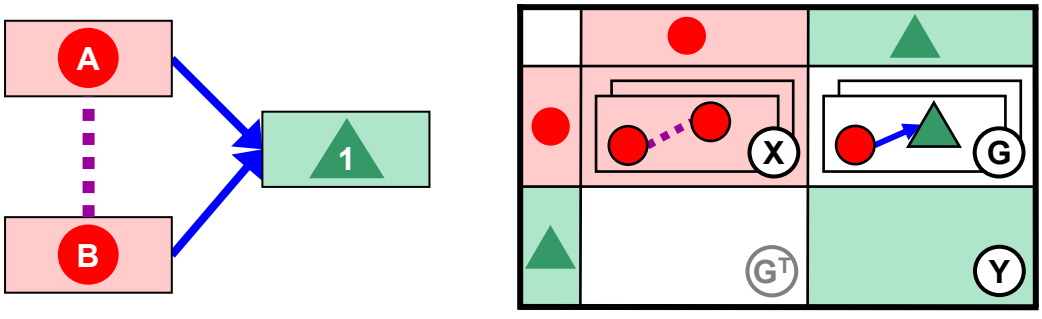


Figure 3-33 Computation scheme for the deduction of a DSM by one DMM (case 1)

$$X = G \cdot G^T \qquad \textit{Equation 3-1}$$

$$Y = G^T \cdot G \qquad \textit{Equation 3-2}$$

Case 2 (Single DMM construction 2): The procedure and obtainable results for this case are for the most part identical to case 1. As can be seen in the graphical representation at the left side, the linking direction between the two domains is inverted compared to case 1. The equations 3-3 and 3-4 show the required computation scheme. For the findings obtained from the deduced DSM X, the statement made for case 1 is also valid. Groups of intensely related elements can be visualized, but only a few analysis algorithms can be applied due to the non-directed dependencies.

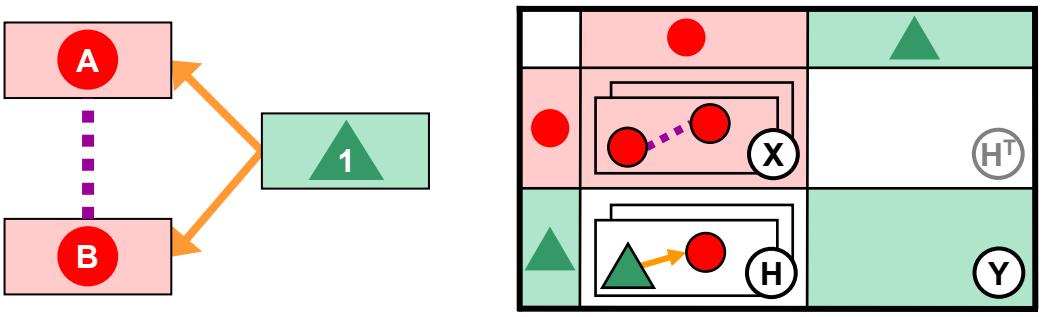


Figure 3-34 Computation scheme for the deduction of a DSM by one DMM (case 2)

$$X = H^T \cdot H \qquad \textit{Equation 3-3}$$

$$Y = H \cdot H^T \qquad \textit{Equation 3-4}$$

Case 3 (Dual DMM construction): In this use case a DSM is derived from available information in two DMMs linking two domains in opposite directions; Figure 3-35 depicts the domain mapping logic. The filled matrices G and H are acquired from information acquisition. The objective matrix X can be obtained by applying equation 3-5; matrix Y can be derived by computing the same two DMMs in inverse order, as is noted in equation 3-6. The resulting networks possess a linking direction, emerging from the linking direction via the second (auxiliary) domain. That means that the derived linking order in Figure 3-35 is directed from element A to element B, according to the elements' indirect connection via the second domain. Due to the implied directions of the deduced dependencies available, DSM analysis algorithms and methods can be applied to resulting matrices.

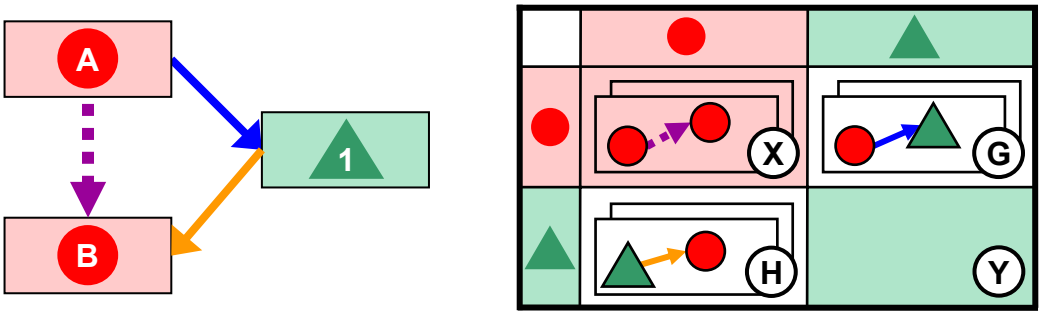


Figure 3-35 Computation scheme for the deduction of a DSM by two DMMs (case 3)

$$X = G \cdot H$$ *Equation 3-5*

$$Y = H \cdot G$$ *Equation 3-6*

Case 4 (Combined DSM and DMM construction 1): Here, the deduction of a DSM is realized by use of dependency information from one DMM and one DSM; Figure 3-36 depicts the related domain mapping logic. Matrix G as well as matrix Y have to be available by information acquisition; the objective matrix X can be obtained by applying equation 3-7; in accordance with case 1 the transformed matrix G^T is required for computation, which can be created without further information.

Matrix Y can be computed by applying the same scheme, if matrix X is available by information acquisition instead of Y. Equation 3-8 shows the required computation, which differs from equation 3-7 only in the sequence of multiplied matrices. The resulting networks possess a linking direction, which results from the linking direction between elements in the auxiliary domain (green triangular domain in Figure 3-36). That means that the linking order in Figure 3-36 is from element A to element B, according to the linking direction between the connected elements in the auxiliary domain.

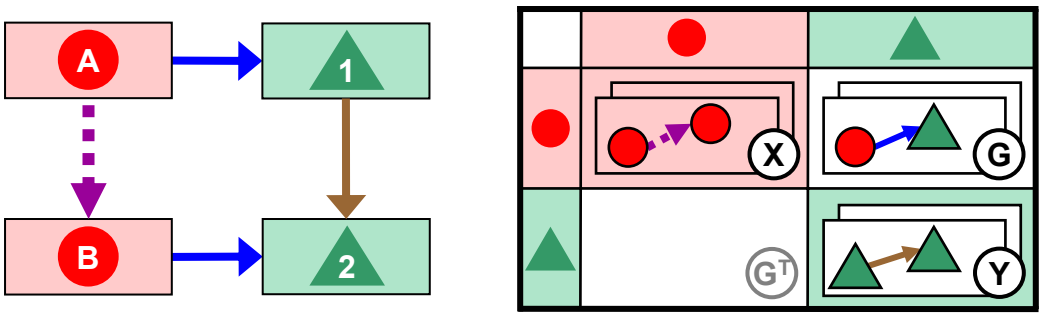


Figure 3-36 Computation scheme for the deduction of a DSM by one DMM and one DSM (case 4)

$$X = G \cdot Y \cdot G^T$$ *Equation 3-7*

$$Y = G^T \cdot X \cdot G$$ *Equation 3-8*

Case 5 (Combined DSM and DMM construction 2): The procedure and obtainable results for this case are generally identical to those described in case 4. As can be seen in the graphical representation at the left side, the linking direction between the two domains is inverted compared to case 4. The linking direction of the resulting network also corresponds to the direction within the linked elements in the second domain. The equations 3-9 and 3-10 show the required computation scheme.

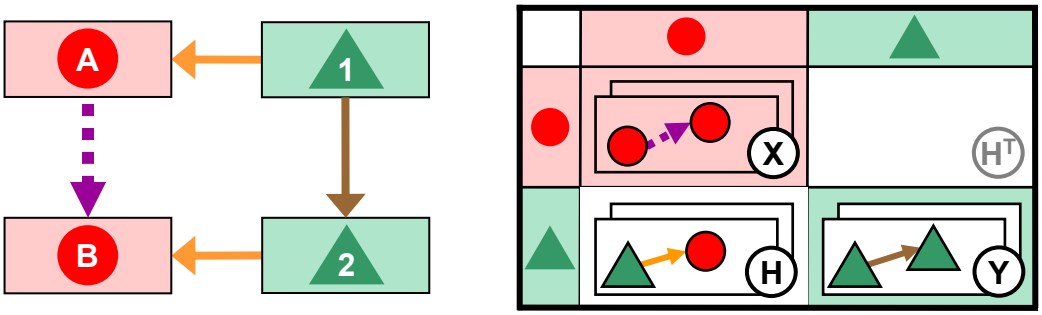


Figure 3-37 Computation scheme for the deduction of a DSM by one DMM and one DSM (case 5)

$$X = H^T \cdot Y \cdot H$$ *Equation 3-9*

$$Y = H \cdot X \cdot H^T$$ *Equation 3-10*

Case 6 (Combined DSM and dual DMM construction): In this case a DSM is derived from available information in two DMMs and one DSM; Figure 3-38 depicts the domain mapping logic. The filled matrices G, H, and Y must be available by information acquisition in order to allow the computation of the objective matrix X. Equation 3-11 shows the related computation scheme. In addition, matrix Y can be computed by use of the same two DMMs and matrix X instead of matrix Y, as is shown in equation 3-12. The resulting networks possess a linking direction, which results from the linking direction of the indirect path connecting the elements of the first domain via the two elements of the auxiliary domain. Consequently, the derived dependency in Figure 3-38 is directed from element A to element B because of the direction of the indirect path via the auxiliary domain.

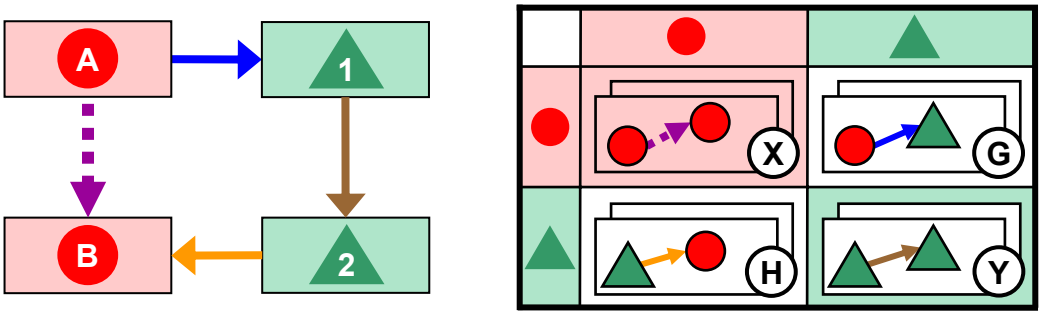


Figure 3-38 Computation scheme for the deduction of a DSM by two DMMs and one DSM (case 6)

$$X = G \cdot Y \cdot H$$ *Equation 3-11*

$$Y = H \cdot X \cdot G$$ *Equation 3-12*

If the computation schemes above are applied to subsets of MDMs, resulting DSMs can obtain numerical values on the diagonal elements. These represent the amount of self-reflexive connections passing indirectly over elements from the auxiliary domain. These values will not be considered in the following analyses of the present work, but may provide potential for further research.

The computation of DSM by information from MDM subsets can be demanding, even if only six different use cases exist and computation schemes are easy to apply. The computational schemes result in six different networks that can be theoretically created for one domain by applying information about its connectivity to a second domain. In a practical application comprising for example five domains, 120 different DSMs can be generated, if information about all sub-matrices of the MDM is available. The following equations show the mathematical determination. Here, the variable D describes the quantity of domains included in the MDM. Equation 3-13 shows that the total quantity of computable DSMs is composed of four types of combinations of MDM sub-matrices. DSMs that can be derived from one single DMM are represented by cases 1 and 2, mentioned previously, and the computation by two DMMs is represented by case 3. DSMs that can be derived from one DSM and one DMM represent cases 4 and 5, and two DMMs plus one DSM are represented by case 6. Equation 3-14 describes the maximum number of derivable DSMs by means of the number of involved domains D. This maximum number only occurs if all sub-matrices of the MDM are filled with information.

$$DSM_{total} = DSM_{1\_DMM} + DSM_{2\_DMM} + DSM_{1\_DMM,1\_DSM} + DSM_{2\_DMM,1\_DSM} \qquad \textit{Equation 3-13}$$

$$DSM_{1\_DMM,\max} = 2 \cdot D \cdot (D-1)$$
$$DSM_{2\_DMM,\max} = D \cdot (D-1)$$
$$DSM_{1\_DMM,1\_DSM,\max} = 2 \cdot D \cdot (D-1)$$
$$DSM_{2\_DMM,1\_DSM,\max} = D \cdot (D-1)$$
$$DSM_{total,\max} = 6 \cdot D \cdot (D-1) \qquad \textit{Equation 3-14}$$

Generally, the computational derivation of DSMs shown can be enhanced by implementing mathematical operators, such as extreme values or logical connectors. This allows for the determination of specific system views, especially if the applied MDM subsets comprise different dependency weights. This increases the achievable quantity of derived DSMs; for example, a dependency in the derived DSM is only established, if the applied indirect dependencies exceed a specific maximum weight. It is not recommended (and nearly impossible) to derive all possible DSMs for further investigations, but rather to carefully select the required ones first due to the quantity of possible computations. This helps keep the amount of analyses to a minimum.

Typically, in practical applications not all sub-matrices of a MDM are filled with information. Thus, the quantity of achievable matrices is lower than the theoretic maximum quantity mentioned before. Nevertheless, it is important not to derive a large number of networks unnecessarily, because this dramatically increases the required effort for further analysis and interpretation. It is highly recommended that the computation of DSM by MDM subsets be carefully planned. Required networks must be defined in the beginning by considering the questions to be solved by network analysis.

Another important aspect when working with MDM computations is the varying data quality in the subsets. If only one of the applied source matrices possesses incorrect information or low data quality, the derived DSM will become inaccurate. Consequently, all input data must be of consistent quality in order to allow their combination for creating derived networks.

3.5.2 Analysis objective

After the provision of DSM structures, the next step in the system analysis is to determine the analysis objectives for the structures considered. This can be the entire network, a subset, or a user-defined combination of subsets. If, for example, developers want to reach a general system overview of network complexity, the entire system will be the analysis objective. However, if specific system components of interest can already be identified, subsets of the structure can be defined for closer investigation. This allows the elements' embedding in their surroundings to be considered and the depiction of their probable integration in specific constellations, such as feedback loops or hierarchical orders. Combinations of subsets can also be useful for structural analysis, especially if users can name several system elements for closer consideration. Such elements can impact the system, because they are often the focus of customization requests from the market. In such cases, developers are interested in determining problematic change propagations in the system, which emerge from adapting the element in question. Also, elements can be specified that cause constraints to the system, for example, because developers do not want or are unable to adapt them. In this case, it might be interesting to know further system elements that are linked to the constricting ones, because their variability in design will be limited as well. Generally, it is recommended that the analysis objective be planned exactly in order to avoid unnecessary or confusing analyses. This can even complicate system comprehension, as it is not the quantity but the suitability of analysis that offers further insight.

3.5.3 Methodical analysis of DSM structures

If the analysis objective of available DSM structures is available, appropriate analysis methods can be applied in order to obtain structural characterizations of the network in question. DSM-related research in particular provides a variety of such methods [BROWNING 2001]. Most of the algorithms refer directly to the matrix representation of networks, i.e., structural characteristics are pointed out by realignment of element orders in matrices. For example, KUSIAK provides sorting procedures in a text-based step-by-step instruction that allows for manual realignment of cyclic and non-cyclic graphs in matrix depiction [KUSIAK 1999]. The advantage of such approaches is that software tools are not mandatory; however,

common spreadsheet software is recommended for reasons of efficiency. Users can already obtain better understanding of the system structure during the execution of the procedure. Nevertheless, manual realignment of matrices can only be applied to small and uncomplicated connected networks, as it is time-consuming. That means that manual procedures are limited to networks comprising only a few elements and dependencies.

A matrix depiction represents only one possible notation of structural content and only provides one possible computation model among others. Its frequent application for network interactions in engineering originates from its compact form of representation and comprehensibility for technical-oriented users. Nevertheless, comprehensive analysis requires the separation of structure mediation for users and notation for access by computational measures. This is clarified in the following example, which shows the analysis of feedback loops by three methods. First, the matrix description of a structure is browsed step-by-step. The second analysis applies a structure description of dependency sets that is sequentially examined. The third method identifies feedback loops by the "Powers of the adjacency matrix". Figure 3-39 shows at the left side a small matrix consisting of seven elements connected by eight dependencies. At the right side the same information about elements and dependencies is given in a description of dependency sets.

	A	B	C	D	E	F	G
A				X			
B							
C	X	X					X
D							
E			X	X			
F							
G					X	X	

$$s(A) = \{D\}$$
$$s(B) = \{\ \}$$
$$s(C) = \{A, B, G\}$$
$$s(D) = \{\ \}$$
$$s(E) = \{C, D\}$$
$$s(F) = \{\ \}$$
$$s(G) = \{E, F\}$$

Figure 3-39 Exemplary structure for discussion of computation approaches

Feedback loop analysis by browsing an adjacency matrix

Obviously, the matrix depiction is more suitable for a user's comprehension of the structure; however, computational analysis can become complicated when executed by use of this representation form. This can be seen by an example of a feedback loop analysis that applies a step-by-step browsing of the matrix depiction in the following:

Step 1 Select first matrix row

Step 2 Search for dependencies in the selected row

Step 3 Search for dependencies in the rows that correspond to the found dependencies

Step 4 If dependencies exist that link on the element that corresponds to the selected row

→ feedback loop

Step 5 Back to *Step 3* until no more dependencies can be found

Step 6 Select next row (until all matrix rows are browsed) and go back to *Step 2*

The procedure represents a meta-code that could be implemented, for example, as a macro in common spreadsheet software. Here the matrix is used as two-dimensional data storage. The algorithm systematically follows all dependency chains by scanning the matrix rows and columns. A feedback loop is identified if such a dependency chain reaches the initial element again. Even if the algorithm works correctly, it is somewhat inefficient and therefore can not be recommended for application to large-scale structures of numerous elements and dependencies.

Feedback loop analysis by description of dependency sets

A more efficient computation procedure that is based on a description of dependency sets is presented in Figure 3-40. Dependency chains are also scanned sequentially. However, this procedure could be implemented more efficiently as no (time-consuming) switching between rows and columns is required and paths already known do not have to be iteratively examined. In Figure 3-40 the dependency set of element A is first considered.

$$
\begin{aligned}
&s(A) = \{D\} \\
&s(D) = \{\ \} \quad \Big\} (1) \\
&s(B) = \{\ \} \\
&s(C) = \{A, B, G\} \\
&\qquad s(A) \rightarrow (1) \\
&\qquad s(B) = \{\ \} \\
&\qquad s(G) = \{E, F\} \\
&\qquad\qquad s(E) = \{C, D\} \\
&\qquad\qquad\qquad \Rightarrow C - G - E \quad \textit{feedback loop} \quad \Big\} (2) \quad \Big\} (3) \\
&\qquad\qquad\qquad s(D) = \{\ \} \\
&\qquad\qquad s(F) = \{\ \} \\
&s(D) = \{\ \} \\
&s(E) \rightarrow (2) \\
&s(F) = \{\ \} \\
&s(G) \rightarrow (3)
\end{aligned}
$$

Figure 3-40 Analysis of feedback loops by dependency set notation

The notation s(A) indicates the group of all existing dependencies starting from element A. This group of dependencies only contains the dependency to element D; however, the group

of dependencies starting from element D is empty. Therefore, elements A and D (and their dependency) can not constitute part of a feedback loop. This information can be applied again later in the procedure. The dependency group emerging from element B is empty (no outgoing dependencies from this element), and the dependency group of element C is considered next. First, element C links to element A; this dependency path can be neglected, as dependency chains passing through element A have already been determined as non-relevant for feedback loops. The dependency from element C to element B can also be neglected, as the group of vectors emerging from element B is also empty. Continuing this procedure, it only takes 13 steps to search the entire structure, and the existing feedback loop (linking C, G, and E) can be reliably found.

The equation-based structure representation is not necessarily the most efficient one and is not appropriate for direct implementation to a software tool. However, it permits all existing feedback loops to be determined in the example of the structure by 13 steps, and it only needs to search the entries of the dependency groups sequentially. In contrast, browsing the matrix mentioned before results in switching 53 times between the rows and the columns to examine all feedback loops. Obviously, the matrix representation is more suitable than equations for mediating the structural content, but it is less appropriate for computational analysis.

Feedback loop analysis by the "Powers of the adjacency matrix"

The "Powers of the adjacency matrix" represent a further possibility of DSM-related feedback loop analysis that is described in the literature [DSM INTERNET PAGE 2007]. It provides an adept computational approach by applying matrix multiplication. If a binary DSM[34] is raised to the n-th power, the resulting matrix shows on the diagonal cells' non-empty entries for the elements that can be reached from it in n steps. This means that entry "1" in the diagonal of an element in the third power of a DSM indicates that this element is part of a feedback loop comprised of three elements. Figure 3-41 illustrates the results of three matrix multiplications in an example of a structure that contains four elements (adapted from [DSM INTERNET PAGE 2007]).

	A	B	C	D
A	0	1	0	0
B	1	0	1	0
C	1	0	0	1
D	1	0	0	0

DSM

	A	B	C	D
A	1	0	1	0
B	1	1	0	1
C	1	1	0	0
D	0	1	0	0

Square

	A	B	C	D
A	1	1	0	1
B	2	1	1	0
C	1	1	1	0
D	1	0	1	0

Cube

	A	B	C	D
A	2	1	1	0
B	2	2	1	1
C	2	1	1	1
D	1	1	0	1

Power 4

Figure 3-41 Identifying loops by "Powers of the adjacency matrix" (adapted from [DSM INTERNET PAGE 2007])

The square matrix resulting from the original DSM (left side of Figure 3-41) possesses two entries on the diagonal, which indicate that the elements A and B form a feedback loop.

[34] *A binary DSM can only possess two different values in the matrix cells, e.g., 0 and 1.*

Cubing the matrix results in three entries on the diagonal showing that the elements A, B, and C form another feedback loop. The deficiency of this approach can be seen, if the fourth power of the matrix is computed (right matrix in Figure 3-41). Because of a feedback loop including all four elements of the initial structure (A, B, C, and D), all diagonal cells show non-empty entries; however, the diagonal cells of the elements A and B each contain the value "2". This results because these elements form their own feedback loop (comprising only these two elements), which is run through two times in the fourth power of the matrix. This shows that comprehensive feedback loops of larger structures can not be analyzed by this approach, because feedback loops containing the same elements can not be distinctively traced back to their implied elements. The method only allows for determination of the existence, but not the detailed denomination, of feedback loops. Furthermore, this approach does not always permit the length and quantity of the existing feedback loops to be acquired. If in a matrix, feedback loops have to be identified that span six elements, feedback loops comprising two elements are processed three times and feedback loops comprising three elements are processed two times when the matrix is raised to the 6th power. All these feedback loops are added to the resulting values in the diagonal cells of the DSM, making the identification of a specific feedback loop almost impossible.

Since it has been shown that matrix-based methods possess some disadvantages, a common basis for the description of all DSM analysis methods is given by the graph theory. Generally, the graph theory describes systems comprising elements and their dependencies [BOLLOBÁS 1990]. Usually, elements are indicated as nodes and dependencies as edges. Section 3.4 provides an overview of useful graph visualization techniques; here the aptitude of graphs for the analysis of complex structures will be considered. In fact, all DSM analyses that are based on the realignment of matrix rows and columns can be executed by possibilities of the graph theory, which provides the mathematical basis of DSM analysis. The consequent application of graph theory on DSM-conforming structures not only allows a more efficient application of known methods (e.g., triangularization), but further provides a large set of analysis methods. In the following section, a collection of basic analysis criteria will be presented, which describe network characteristics concerned with graph theory, but which have only been partially applied to the analysis of product development structures so far. Some criteria are already in use in specific methodologies, but they have not been implemented for further suitable analysis purposes. Additionally, methods will be described that have emerged from DSM research and have their origin in the identification of network characteristics by visual realignment of matrix rows and cells. The fact that these methods are also based on general graph theory can help their optimization and improved implementation.

In the following, the criteria have been classified alphabetically and by their objective of analysis. Criteria can aim at the characterization of single nodes or edges (Table 3-2), characterization of subsets (Table 3-3), or characterization of entire systems (Table 3-4). The tables contain a short explication for every method and an example. It must be noted that criteria can possess a different significance in different networks, e.g., product component or process networks. The significance also depends on the chosen dependency meaning. Here the examples represent product component networks linked by dependencies that express change impact between the components. This means that two components are linked, if the adaptation of the first one can ask for a further adaptation of the second one.

Because of highly differing network significance, it is not possible to provide a comprehensive correlation between analysis criteria and their expressiveness. For this reason, the example chosen of a component-based impact network serves only as an illustration; the transfer to a specific network meaning remains to be done by the users. Nevertheless, all criteria are defined and described in detail in the Appendix of the present work.

Basic analysis criteria for the characterization of nodes and edges

The **active sum** mentioned in Table 3-2 quantifies the number of dependencies (in the case of the weighted dependencies, the sum of the dependency weights) that emerge from one node of the structure. The term refers to the direction of impact, i.e., adaptations to an active element result in numerous impacts that change other elements of a structure. Consequentially, a low active sum of an element means that it possesses only a few outgoing dependencies, and therefore there is less impact to further elements if it is adapted. A rule of thumb for deriving measures from the criterion of the active sum in practice is the avoidance of adaptations to product components possessing high active sums or determination of such components early in the development process. The later such an element is specified, the more rework can result to the system due to the high amount of possible change impacts. The active sum criterion is applied, for example, to product development methods in the form of the influence matrix and related influence portfolios ([GOMEZ & PROBST 1997], [ZANKER 1999, P. 24], [AMBROSY 1996, P. 52]).

The **activity** is computed by dividing the active sum by the passive sum (see the definition below). It presents an informational accumulation of these two other criteria and is applied to the influence matrix and influence portfolio. Constant activity values can be visualized in an influence portfolio by straight lines through the origin.

An **articulation node** depicts a bridge between subsets of a structure. This means that impact from one subset to the other must pass through this node. This structure criterion is applied in methods of process planning, where articulation nodes represent bottlenecks. Several software tools for process planning provide algorithms for detecting such articulation nodes in comprehensive networks. However, this criterion has not been applied to methodical product component modeling so far.

The **attainability** of a node represents an index that indicates to which extent a specific node can be reached by other nodes of a structure. This criterion is related to the analysis criterion of closeness (see the definition below) and is based on the research of FREEMAN, who applied the criterion for characterizing communication networks [FREEMAN 1979]. The higher the value of attainability of a node, the more other nodes can reach it, or it can be reached by dependency paths.

A **bridge edge** depicts the counterpart of an articulation node, but describes an edge connecting two subsets of a structure (instead of a node). The application of bridge edges can be found in process and information flow networks, and algorithms for their identification are implemented in related software tools.

Table 3-2 Basic analysis criteria for the structural characterization of nodes and edges

Analysis criterion	Explanation	Illustrative application
Active sum	Quantity of outgoing edges	Component with high active sum provides numerous impacts to further components
Activity	Division of active sum by passive sum	The activity shows a component's degree of active participation in change impacts
Articulation node	Only node connecting two subsets	Every change impact between subsets must pass by the component
Attainability	Node can be reached from other nodes by edge paths	Component with high attainability receives/ provides impact from/to many further components
Bridge edge	Only edge connecting two subsets	Change impact between subsets must pass by the dependency (the change impact of the edge takes part in many dependency chains)
Bus	Node possessing a multitude of incoming and/or outgoing edges	A connecting component for many other components, e.g., the frame of an automobile
Closeness	Specifies the distance from/to other nodes in the graph	Component with a high closeness receives/provides direct impact to many further components (short distances by dependencies)
Criticality	Multiplication of active sum and passive sum	The criticality shows a component's degree of integration to change impacts in the system
End node	Node possessing only incoming (passive) edges	Component only (passively) receives change impacts from the system
Isolated node	Nodes without any edges to other parts of a system	Component is not affected by and does not provide any change impact to the system
Leaf	Node possessing edges to one other node only	Change impact to or emerging from a leaf can only result from or affect the component directly connected to the leaf component
Passive sum	Quantity of incoming edges	Component with a high passive sum receives numerous impacts from further components
Start node	Node possessing only outgoing (active) edges	Component only (actively) spreads change impact to the system
Transit node	Node possessing one incoming and one outgoing edge	Change impact is only passed by the component from one passively and one actively connected component

A node possessing either a multitude of incoming or outgoing dependencies represents a (active or passive) **bus**. Busses can be easily detected in matrix depiction. Active busses possess many dependencies in the rows, whereas passive busses possess these in the columns. YU ET AL. describe an algorithmic approach that automatically aligns busses in exposed

positions close to the borders of the matrix [YU ET AL. 2005]. An element representing an active bus of a component structure must be specified as early as possible in the development process, because a multitude of further elements can be impacted by this component. In contrast, an element representing a passive bus can be seen as an integrator that receives impacts from many other components. Consequently, its specification must be downstream in the development process in order to avoid multiple rework on this component.

The **closeness** describes the distance of a specific node from all other nodes in a system measured by the length of the dependency paths and accumulated in an index. The closeness criterion was first used for the characterization of communication networks [FREEMAN 1979]. Nodes that possess a high value of closeness (the definition in the Appendix allows a maximum value of 1) are linked only by short dependency paths to many other nodes in a structure. In combination with the criterion of attainability, this criterion provides information about the central positioning of a node in the structure and its reachability.

Like the activity criterion already mentioned, the **criticality** of a node represents an accumulation of information from the criteria active sum and passive sum. The criticality is computed by multiplying the active and the passive sum of a node and is typically applied in influence diagrams. In these diagrams, lines of constant criticality are represented by hyperbolas symmetrical to the bisecting line of the portfolio area. A high criticality value means that the node will likely be involved in adaptation processes of the system. A node possessing a high criticality can either spread change impact to many other nodes due to a high active sum, or it can be affected by other nodes due to a high passive sum. Of course, a high active and a high passive sum results in the highest criticality value of a node. Medium values for the active and passive sum result in a relatively high criticality value as well (comparable to the combination of a high and low active/passive sum). In this case, the node can influence other nodes and can be influenced by other nodes in equal measures.

End nodes only possess incoming dependencies, i.e., an end node represents a fully passive element of the structure that does not influence any other elements. This criterion is used indirectly in influence matrices and portfolios, because the value of the active sum of end nodes is zero. Available algorithms of DSM analysis typically align end nodes in matrices as the last elements (in the lowest rows and columns furthest to the right) [KUSIAK 1999, P. 35FF]. In product structure analysis, the characterization of a component as an end node suggests its specification late in the development process, because any impact from this component to others will not occur; its adaptations can not provoke any change impact to other components.

An **isolated node** does not possess dependencies to other nodes of the structure considered. Impacts to the isolated node do not affect the rest of the structure or, conversely, user-defined system adaptations do not affect the isolated node. For this reason, an isolated node can normally be excluded from further considerations of the structure. Considerations of isolated nodes can be investigated by plausibility checks on the structure. Isolated nodes often emerge from the combination of structures that have been separately generated. This can, for example, indicate a possible lack of interface definitions, but it depends on the meaning of elements and dependencies specifically considered. If, however, the structure has been

designed correctly, the identification of isolated nodes helps concentrate on relevant parts of the structure.

The criterion **leaf** indicates a node, which is only connected to one other element in the structure. A further classification into active, passive and bi-directional leaves is possible. In a component-based model, a leaf is easy to handle, as it only affects one component directly (active leaf) or becomes affected by only one specific component (passive leaf). Component adaptations to the leaf and the directly linked component can therefore be considered simultaneously. The existence of a leaf accompanies the existence of an articulation node and a bridge edge because of the single connector between the leaf and a further element of the structure.

The **passive sum** quantifies the number of dependencies (in the case of weighted dependencies, the sum of the dependency weights) that affect one node of the structure. The term refers to the direction of impact, i.e., a passive element tends to obtain impact from numerous other elements of the structure. A low passive sum of an element indicates only a few incoming dependencies, and therefore there is less probability of impact to this element if there are adaptations to further elements of the structure. Generally, elements possessing a low passive sum should be specified early in the development process, as they will not acquire numerous change impacts later on. The passive sum criterion and the active sum criterion are applied simultaneously in product development methods, e.g., in influence matrices and related influence portfolios.

The **start node** criterion represents the counterpart of the end node criterion already described and characterizes elements that do not possess incoming but only outgoing dependencies. A start node is a fully active element of a structure. Available algorithms of DSM methodology typically align start nodes in matrices as the first elements (in the first rows and leftmost columns). In component-based system structures, a start node represents an element that should be specified in the beginning of the development process. As this node is not affected by any other system elements, no rework due to change impact can occur.

A **transit node** is characterized by only one incoming and one outgoing dependency; it can often be combined with one of the two directly linked nodes (one actively and one passively connected element), and then excluded from further consideration for the structure modeling. If this procedure can be applied (depending on the specific network meaning), this simplifies further analyses, because the quantity of elements can be decreased without exclusion or falsification of relevant structural constellations.

Basic analysis criteria for the characterization of subsets

Besides analysis criteria concerning single nodes and edges, those related to subsets of the structure in question are of particular interest. Often they require more computational efforts for their identification than the criteria mentioned before. That means that software support is required for effective application in most use cases. Table 3-3 presents a collection of useful analysis criteria related to structural subsets that will be explained in the following section. Detailed definitions and possibilities of application are presented in the Appendix.

A **bi-connected component** describes the subset of a graph, where all nodes are linked to each other (not necessarily by direct dependencies) and the removal of one user-defined dependency does not break up the general connectivity of the subset. Thus, in a network that represents product components, none of the dependencies represents an exclusive connection to specific network parts or elements. Consequently, the existence of bi-connected components are desired for reasons of system reliability (removal of one dependency does not completely impede the interaction between the elements of the bi-connected component); however, bi-connected components can be opposed to modular design, if a bi-connected component stretches across the planned parting line of two modules. In this case, the independence of subsets that have to be modularized can only be realized by multiple measures to the system (avoidance of at least two edges). By contrast, only one dependency would have to be removed for modularization, if an articulation node or a bridge edge crossed the parting line of the planned module subsets.

Generally, **clusters** represent structural subsets that possess a high amount of internal dependencies compared to their number of external dependencies. If the elements of a cluster are aligned side by side in a matrix depiction, their dependencies form a square block along the diagonal. Only in complete clusters are all elements mutually linked and the matrix block entirely filled with dependencies. Clusters can also be defined by means of the criterion's strongly connected part (see definition below). In this case, all elements that create part of a strongly connected part form the cluster. Whereas in complete clusters all elements are mutually linked by direct dependencies, the elements in a cluster based on a strongly connected part are at least all connected indirectly.

The **distance** criterion is highly related to those of closeness and attainability, which have been described above. In contrast to these, distance describes the minimum number of dependencies (in the case of weighted dependencies, the minimum sum of dependency weights) between two nodes. That means that the distance is counted along a dependency chain between two nodes, taking into consideration the direction of the dependencies. In a component-based structure, short distances mean direct impact between nodes. The longer a distance becomes, the more likely that adaptations to the initial node will not affect the second one due to moderation effects. However, long distances can also become dangerous in product development because the resulting possible effects are difficult to predict for users.

Feedback loops describe two or more elements that interlock and influence each other reciprocally. This can provoke considerable impact on the entire system, as in a component network, feedback loops can provide self-energizing or self-impeding effects once a specific element is adapted. Such effects are often difficult to control, especially if several feedback loops are mutually overlapping. In this case, one element change can activate a multitude of effects resulting from different feedback loops. It depends on the specific use case, if short or long feedback loops are more difficult to handle in product development. On the one hand, short feedback loops can result in fast, self-energizing effects. On the other hand, this effect proceeds more slowly in long feedback loops; however, the existence of long feedback loops is often hard to detect for product developers. Matrix-based algorithms for feedback loop detection are available in methods of product development (see Figure 3-41) [KUSIAK 1999, P. 36FF]. However, these approaches fail in cases of comprehensive structures comprising

numerous feedback loops that overlap each other in nodes and edges. As will be shown later in Section 3.6 and Chapter 4, knowledge about that overlapping in particular provides possible effective structure optimization.

A hierarchical subset of a structure is characterized by a dependency chain starting from one top element and expanding to a multitude of further elements over several levels. A **hierarchy** subset does not contain dependencies reaching from subordinated elements back to superior ones. In a component-based impact network, the element that is on the highest level of a hierarchy can induce a whole cascade of change impacts on all subsequent components, if the top level element is adapted. Consequently, elements on the lower levels of the hierarchy are better suited for changes than top level elements. In practice, it is difficult to detect hierarchical subsets within a complex product structure manually, and appropriate software for computation and visualization is required. In DSMs, elements can be aligned in a way that hierarchies appear in staged (see Figure 3-24) or triangular (see Figure 3-23) form. A staged form results if the hierarchical subset represents a tree structure without any cross-linking between the branches. Like the criterion of feedback loops, hierarchical subsets can occur in high quantity and overlap each other in practical applications.

The **locality** depicts the surroundings of a specific element, i.e., all nodes that are directly linked with the node in question. The active and the passive locality, which comprises the outgoing and incoming dependencies of a node respectively, can be differentiated. The locality criterion is, for example, applied in the TouchGraph graph layout and is meant to focus on specific aspects of a structure [TOUCHGRAPH INTERNET PAGE 2007]. It further allows easy browsing in the structure by moving from one locality to neighboring ones. The locality is helpful for evaluating the structural embedding of specific nodes and helps identify their behavior in the system. Locality observations are not limited to the directly-linked nodes, but can be extended, for example to all nodes that can be reached in two steps.

A structure analysis often applied identifies a **path** between two nodes in question. Possible specifications represent the shortest and longest paths as well as paths fulfilling specific constraints (e.g., passing by an exactly defined third node). A path analysis is not suitable to be executed in matrices and is typically done by means of graph theory descriptions. Process oriented modeling, in particular, such as network planning techniques and critical path analyses, apply appropriate methods and algorithms. Furthermore, communication networks make use of path analysis to evaluate human relationships [FREEMAN 1979]. This structure criterion is required for applying trace back and feed forward investigations (see Section 3.6).

The **quantity of indirect dependencies** has been applied by MAURER ET AL. for plausibility checks on the dependencies acquired by interview [MAURER ET AL. 2006]. This criterion can identify nodes that depend highly on each other, but often do not possess a direct dependency. To what extent indirect dependencies will be considered has to be defined. Typically, dependency chains passing by one or two further elements are most relevant. For example, when modeling component-based structures, two components may not possess a direct dependency, but may be connected indirectly by dependencies to numerous other components. If such component pairings can be identified, both elements are assumed to be considered simultaneously in case of product adaptations; it is highly likely that adaptations to one of the elements will affect the second. A matrix depiction is not suitable for computing

the indirect dependencies; however the representation of this criterion can be intuitive in matrices (visualized in the Appendix).

Table 3-3 Basic analysis criteria for the structural characterization of subsets

Analysis criterion	Explanation	Illustrative application
Bi-connected component	Subset where the elimination of one edge does not separate the structural coherence	Components are intensely connected and can not have been separated by one single measure
Cluster	Subset contains a large number of internal edges compared to external ones	Components of a cluster are suitable for declaring a module, because adaptations to the implied components will not cause numerous change impacts to further external components
Distance	Specifies the minimal number of edges between two nodes in a structure	Components with a high value of mutual distance indicates only an indirect impact between them
Feedback loop	Subset with circularly arranged edges	Change impacts affect the originating component via further components in a feedback loop
Hierarchy	Node branching out by numerous edges into further nodes on several levels	Change impact to a component that forms the top component of a hierarchy can provoke change impact to many further components
Locality	Subset that includes the surrounding nodes of a central node because of directly connected edges	Component can provide direct change impact to all components directly connected by outgoing edges
Path	Connection between two nodes by edges	The shortest dependency chain (path) between two components can represent the most probable change impact between them
Quantity of indirect dependencies	Quantity of edges between two nodes that pass by further nodes	Components that are frequently linked by indirect dependencies will influence each other in all probability if one of them is adapted
Similarity	Two nodes possess a high quantity of identical nodes that are directly linked to them	A high similarity of components assumes their similar behavior in the case of occurring change impacts to a system
Spanning tree	Subset connecting all system nodes by a selection of existing edges	The minimum spanning tree indicates the minimum of change impacts that can occur without breaking up the system coherence
Strongly connected part	All nodes can be mutually reached by a edge path	Every component possesses a direct or indirect change impact to any other node in the subset

The criterion of nodes' **similarity** is indirectly applied in some methods, such as clustering or sequencing [McCormick et al. 1972], and describes the degree of identical impact of specific nodes to or from other elements of the structure. It is further applied to DMM analysis, where it is used to determine a suitable alignment of nodes [Danilovic & Browning 2004]. Nodes' similarity can provide useful insight, especially if further structure criteria are difficult to identify. If required, the criterion can be further classified as active and passive by comparing only outgoing or incoming edges, respectively. Additionally, similarity can refer to existing, as well as non-existing, edges. If, for example, in a component-based network two nodes possess a high similarity in their outgoing edges (active similarity), both cause similar impact to the system if they are adapted.

A **spanning tree** represents a subset of the network in question containing all nodes, but only a minimum of the existing edges required for retaining the coherence of the entire graph. Relevant specifications of the criterion are minimum or maximum spanning trees that take the weight of edges into account for selecting the edges of the subset. Spanning trees describe the core structure of the system in question, which is at least required without dissolving its integrity. Generally, several spanning trees can be identified in a structure.

Strongly connected parts characterize a subset, where every node can be reached by all others by at least one dependency path. This means that strongly connected parts represent an enhancement of the cluster definition, because all nodes of the subset have to be linked to each other, however not necessarily directly. In a component-based system structure, a strongly connected part describes a collection of components that can all mutually affect the others by change impact if adapted. Consequently, strongly connected parts are suitable for defining the extent of planned modules or building blocks.

Basic analysis criteria for the characterization of systems

The third group of basic analysis criteria focuses on the characterization of the entire structure. Almost all of these criteria apply matrix visualizations, even if some can also be displayed in other forms. The criteria presented in Table 3-4 make also use of criteria that aim at the characterization of nodes (e.g., start and end nodes) or subsets (e.g., similarity or strongly connected parts), which have been presented before. If comprehensive systems in particular are examined, software support is required, because computational effort increases rapidly with the number of nodes and edges involved. Step-by-step manuals are available for the application of some criteria ([Warfield 1973], [Kusiak 1999, p. 29ff]); they help users obtain a better general system understanding, but do not allow for practical analyses if system structures become large.

Banding represents an enhancement of the partitioning analysis (see the explanation in Table 3-4 and definition below) and is meant for indicating mutually independent nodes in a structure by realignment of elements in the matrix depiction and subsequent application of light and dark bands to the matrix [Grose 1994]. If those elements are located side by side and can be processed independently from each other, they are assigned the same band. Banding refers to the different hierarchical levels identified by partitioning a matrix, but the occurrence of feedback loops is ignored. For this reason, the useful applicability of banding to complex systems can be doubtful, as the analysis is based on neglecting important structural

characteristics. The criterion is designed for the analysis of process or activity networks. Here the objective is to obtain as few bands in a matrix as possible, resulting in a high degree of parallelism (concurrency) for the process steps. If several elements exist in a band, one represents the critical element, which determines the minimum required resources. As mentioned already, the non-consideration of feedback loops in this analysis criterion raises doubts as to its applicability, because process or activity networks are often affected by iterations in practical use cases.

The **clustering** criterion allows the identification of specific subsets of a structure; however, here it is assigned to the criteria of entire systems, because the existence and the quantity of clusters characterize the system itself. The objective of the clustering analysis is to detect subsets that possess many internal dependencies and as few dependencies as possible to further external nodes of the structure. Consequently, in a component-based network, a cluster represents an appropriate basis for creating a product module. An adaptation to one element of the cluster causes many impacts to other elements in this subset, whereas the replacement of the whole cluster (module) only requires a few dependencies to be considered for other product parts. PIMMLER & EPPINGER published an example of a DSM-based clustering analysis for an automobile Climate Control System [PIMMLER & EPPINGER 1994]; however, the chosen network consisted of only 16 elements. Only one of the three identified clusters overlapped with the two others, which allowed the depiction of all clusters in a DSM. Further research on DSM clustering is available from FERNANDEZ or THEBEAU ([FERNANDEZ 1998], [THEBEAU 2001]). As has been shown in Figure 3-24 and Figure 3-25, DSMs are limited in their representation of overlapping subsets. If more than two clusters mutually coincide, other visualization forms have to be adopted. Graph theory provides several clustering algorithms [HARTIGAN 1975], some based on the application of the similarity criterion (see Table 3-3 and definition above). In addition, clusters can be identified based on the criterion of strongly connected parts (see Table 3-3). As all nodes included in a strongly connected part possess at least one path extending to all other nodes of the subset, the internal connectivity is necessarily higher than the external. Approaches for DSM-based clustering exist that apply genetic algorithms, for example, by using the Minimum Description Length (MDL) presented by YU ET AL. [YU ET AL. 2003].

The **degree of connectivity** represents a criterion that is easy to apply to user-defined structures. The ratio of existing edges to the quantity of all possible edges can serve as plausibility check and help the selection of appropriate analysis criteria as well as the selection of the level of detail. For example, as similar products typically possess similar structures, a comparison of the degree of connectivity can be an initial indication of the available data quality after information acquisition. A high degree of connectivity can make the application of specific analysis impossible; for example, the quantity of feedback loops can increase significantly with higher connectivity of elements in the structure. Even if software support provides computational possibilities for their identification, this generally does not permit further interpretation. A high degree of connectivity can indicate that the information acquisition was executed at a too abstract level. If, for example, an entire automobile were described by only four components (power train, car body, chassis, and interior equipment), all elements would have to be mutually linked in an impact network. Obviously, this does not allow any structural analyses and does not provide significant

insight. As with increasing level of detail, the quantity of system elements increases as well, a compromise has to be found to consider the right level of detail, the degree of connectivity, and a manageable quantity of system elements.

Table 3-4 Basic analysis criteria for the structural characterization of systems

Analysis criterion	Explanation	Illustrative application
Banding	Enhancement of partitioning, identification of elements that can be executed in parallel or sequentially	Identification of an optimized hierarchical order of components allows for the determination of a process sequence for minimizing iterations caused by change impact
Clustering	Identification of subsets with a high internal amount of edges (compared to external links to the surrounding)	Clusters are appropriate for module building because adaptation of one component of the cluster often causes cluster-internal change impact
Degree of connectivity	Percentage of existing edges compared to the theoretically possible quantity of edges	An exceptionally high or low degree of dependency suggests deficiencies in information acquisition
Distance matrix	Specifies the distances between all node parings in a structure and represents them in matrix depiction	Ordering component blocks in the matrix by length of distance identifies the degree of change impact between component groups
Matrix of indirect dependencies	Specifies the quantity of indirect dependencies between every node pairing	Plausibility check during information acquisition: many indirect change impacts between components can suggest the existence of a direct one
Partitioning, triangularization, sequencing	Sequential or block order of nodes	Best adaptation sequence concerning effects of change propagation; minimization of iterations due to backward change impacts

The **distance matrix** applies the distance criterion already described (see Table 3-3) and depicts the resulting values for all node pairs in the cells of a DSM. The rows and columns of the matrix can be realigned in order to build blocks of similar distance values. For smaller matrices, these can be executed manually; however for larger matrices cluster algorithms are recommended. In a component-based distance matrix that is aligned by blocks of similar distance values, product developers can identify element groups that are closely related or, in contrast, are only connected by far-reaching, indirect dependency chains. Thus, in case of required component adaptations, developers can focus on relevant groups of elements for evaluating the possible change impact.

The application of the **matrix of indirect dependencies** provides an efficient possibility for plausibility checks during information acquisition as well as for obtaining a general system understanding. Here, the analysis of indirect dependencies is applied to every node pairing of the structure, and accumulated quantities are depicted in the cells of a DSM. Case studies

have shown that in component-based networks, the existence of a dependency often accompanies a high quantity of indirect dependencies between both elements [EICHINGER ET AL. 2006]. This experience can be used to specifically point out node pairings for closer consideration in the information acquisition process. Additionally, in the information acquisition process, people tend to declare direct dependencies to nodes, which are only indirectly linked in reality ("everything is linked to everything"). The matrix of indirect dependencies permits a better system understanding, as indirect dependencies are generally difficult for developers to identify in existing structures.

Partitioning describes the reordering of a DSM (its rows and columns) with the objective of arranging all existing dependencies at one side of the diagonal. If such an alignment can be found, no feedback loops exist in the structure. However, partitioning is not an appropriate criterion for feedback loop analysis (see Figure 3-26 and Figure 3-27). On the one hand, more efficient algorithms exist for determining the non-existence of feedback loops. On the other hand, if the application of partitioning results in dependencies that constitute part of the feedback loops, the quantity and formation of the loops generally can not be further specified (it should be mentioned that strongly connected parts of the considered structure can be specified by partitioning). However, in process or activity networks the matrix resulting from the application of partitioning indicates an appropriate sequence of process-step execution. Consequently, the identification of a matrix alignment with all dependencies at one side of the diagonal identifies a work flow without any required iteration steps. Often complex structures possess feedback loops that do not allow for such an ideal alignment. Then partitioning tries to align a minimum of edges below and all edges as close as possible to the diagonal; i.e., feedback loops are kept at a minimum length. This optimization only makes sense in time-based structures, as feedback loops of minimum length mean time savings. Other expressions for partitioning are **sequencing** or **triangularization**. Several related algorithms are described in the literature, from KUSIAK ET AL., KUSIAK, and GEBALA & EPPINGER ([KUSIAK ET AL. 1994], [KUSIAK 1999, P. 35FF], [GEBALA & EPPINGER 1991]). However, these algorithms do often not provide optimal results, as overlapping feedback loops can not be satisfactorily processed. A prospective possibility of amelioration is the combination of common deterministic algorithms with permutations. Pre-classification can be done by established algorithms, whereas highly interrelated areas of the structure (comprising the overlapping feedback loops) can be systematically permutated for obtaining optimized alignments. If the quantity of elements is limited for applying the permutation, computational effort can be handled efficiently.

Further analysis concerning MDMs

All analysis criteria mentioned here refer to the exploration of DSM matrices, i.e., structures containing one domain and one dependency type only. It must be noted that DMM structures connecting two domains can also be analyzed by basic analysis criteria, if such networks still comprise only one dependency type. This seems to be helpful, as DMM networks can be extracted from MDM models by applying matrix multiplication similar to the procedure presented for deriving DSM networks. In addition, DMM structures are often easier to obtain in information acquisition [KUSIAK ET AL. 2006]. Applicable criteria concern the characterization of subsets or the entire system. For example, DANILOVIC & BROWNING apply

the similarity criteria to such DMM networks in order to achieve clusters [DANILOVIC & BROWNING 2004]. However, DMM analysis has not been much addressed in the literature so far. Even if the criteria presented seem to be helpful, the significance of the results and possible interpretations have to be examined in future work. Furthermore, the DMM networks make new demands on visualization techniques which provide intuitive understanding and possibilities of user interaction.

Reflections on structure analysis

The basic analysis criteria for DSM structures allow for the characterization of complex structures and their elements, and therefore represent useful methods for obtaining better system understanding and important input for further system optimization. However, the application of DSM analyses in a MDM environment also poses some challenges and difficulties, which have to be considered. Initially, a number of possibilities exist for the extraction of data from a MDM system, which is applied as an analysis base. In addition, a large set of analysis criteria are on hand. That means that in most use cases it will be impractical to apply all possible analyses to all derivable data sets, but a suitable selection is mandatory. Unfortunately, the significance of analyses is generally case specific, i.e., the aptitude of a selection of an analysis method once made can only be rated after its application. In order to avoid multiple rework, experience in structural analysis is highly recommended for the implementation of DSM analyses, because rules of thumb have not been explicitly available. Besides the quantity of derivable DSMs and applicable analysis criteria the specific network content does highly affect the significance of applied analyses. This becomes even more complicated, when the network meaning results from a combination of up to three DMM and DSM subsets.

It is important that single analysis results are considered in the context of further structural characteristics. A brief example can clarify this. If on a specific product structure only the feedback loop criterion is applied, analysis results can show an extremely high quantity of existing feedback loops. With only this information on hand, developers could misinterpret the situation and initiate wrong optimization measures. As a high quantity of feedback loops is usually an indication for difficult product adaptation due to unmanageable change impact, optimization measures would focus on breaking up as many impact loops as possible. However, if the product in question is highly modularized and all feedback loops only occur inside the modules, feedback loops do not present the assumed significance. In structural depiction, modules form clusters that typically possess a multitude of feedback loops due to their high mutual linking of included elements. In fact, a cluster consisting of six completely interrelated elements already forms 409 feedback loops. Nevertheless, feedback loops inside a product module are typically irrelevant for further optimization, as modules are only exchanged en bloc without further internal adaptation. That means that the relevant dependencies in case of product adaptations would be the interfaces between modules and further product components. In the use case example, an additional cluster analysis could show the specific non-relevance of feedback loops and would help avoid misleading interpretation.

The example above shows that general conclusions can not be drawn from the application of single analysis criteria and that a more comprehensive system understanding is required for the interpretation of complex structures. Generally, it should be possible to apply all available analysis criteria to the system in question; however, this would result in extremely large amounts of data that can not be interpreted and therefore are inapplicable for managing complexity.

The fact that the system structure does largely affect the system behavior means that conversely similar products possess similar structures. With increasing experience developers can better rate system structures and derive useful measurements for optimization by a combined use of the analysis criteria presented.

3.6 Discussion of practices

The application of basic analysis criteria to network data provided results in knowledge about structural attributes and characteristics and has been explained in Section 3.5. This can help users improve system understanding and predict system behavior due to applied engineering changes. It is clear from the presentation of analysis criteria that practical measures on system structures need to consider these criteria in combination and that their isolated application can be misleading. With the analysis results on hand, the next step in the methodical structure management approach is to apply the findings for managing a system's complexity better. In order to enable a practical application, the compilation of a structure manual is presented in this section. This approach is meant for situations where an existing product can not be generally redesigned, but frequent requests for adaptation have to be implemented. In this case, product developers require, for example, information about possible change impact due to planned adaptations. The knowledge about strongly or weakly interlinked areas of the structure allows for the prediction of an adaptation effort that has to be expected (and considered in related resource planning). Specific structural subsets can prohibit any changes to elements, because the resulting change impact and therefore adaptations to further elements become too extensive to be executed efficiently. For example, a change to the central frame of an automobile can probably impact a multitude of other components. Therefore, developers can decide not to allow changes to the frame, if adaptation time and costs would be unacceptable. Additionally, some subsets can point out the most suitable system elements for conducting product changes, as resulting change adaptations remain minimal and can be controlled by product developers. Thus changes can be quickly implemented at low costs. A structure manual can provide a scenario, when the main layout of a product exists as a standard base and customer-specific variants have to be derived. In these cases the general structure remains, but developers need to know about probable consequences for the specifically created product due to these adaptations. However, the fundamental structure of the product remains (and depicts again the basis for future variant design).

Whereas a structure manual focuses on improving the interaction with existing product structures and their required specific adaptations, the fundamental optimization of a system structure represents the second practical application to be considered. Structural optimization is helpful when an enterprise wants to redesign the fundamental structure of a product in order to better meet the requirements. Once a system structure is optimized, it can be used to create

a structure manual, which then allows the creation of specific system variants. For this kind of application scenario, three approaches are presented in this section. First, the tearing approach is explained, which represents a DSM-based structure optimization, which is mainly meant for improving process structures. Afterwards, the structural ABC-analysis is presented, which allows dependencies with significant impact on the occurrence of selected basic analysis criteria (see Section 3.5) to be determined. The third optimization approach uses an evolutionary algorithm (EA) for proposing possibilities to decrease the overall system criticality.

3.6.1 Structure manual

The structure manual can support communication in development teams, especially if people from different disciplines are integrated. The provision of visualized networks that clarify domain-spanning dependencies can help developers understand impact chains resulting from their own system interactions that cause change adaptations beyond their scope of responsibility. Experts are normally well informed about the direct dependencies concerning their own sub system; however, far-reaching impact chains are difficult to detect in complex systems, especially if a disturbing impact occurs in a development discipline different from the initiating system change. In terms of reliable product development, it is useful to systematically analyze possible consequences (for example, as it is done by the FMEA[35] methodology), even if a specific impact chain does not produce a negative impact in a specific application scenario. The structure manual can also provide information about opportunities and restrictions for product adaptations. If several possible adaptations can alternatively fulfill a specific product requirement, developers can choose the most appropriate one by balancing required efforts.

A structure manual is comprised of selected information about the occurrence of basic analysis criteria for further interpretation. Results of methodical analyses are systematically presented and facilitate the planning and tracing of system adaptations as well as their resulting impact. These methodical analyses are described in the following text. Table 3-5 lists the methods together with a short explanation and an application scenario example. Further explications are shown in the Appendix.

The **feed-forward-analysis** depicts possible chains of change impact that originate from the adaptation of one specific node. This represents an extension to the impact check list explained below, because not only the direct surrounding of a specific node, but sequences of dependencies are considered. Because of the large amount of possible dependency chains in practical applications, it is nearly impossible to depict and consider all of them. Further constraints, for example, important elements that have to be included in dependency chains, are required to obtain a useful selection. The feed-forward-analysis allows the estimation of consequences of adaptations that may normally remain hidden due to far-reaching indirect dependency effects.

[35] FMEA: Failure Mode and Effect Analysis

The **impact check list** represents a methodical analysis providing information about the nodes directly linked to one specific node in question. Depending on the use case, the impact check list can comprise all nodes connected by outgoing or incoming dependencies only. The provision of this information supports the systematic step-by-step evaluation of impact propagation resulting from the adaptation of a specific system element. If an adaptation has to be executed, product developers can sequentially evaluate the probable occurrence of change impacts to further elements; the systematic provision of depending elements guarantees that no possible impacts are neglected.

Table 3-5 Methods for the compilation of a structure manual

Method	Explanation	Illustrative scenario
Feed-forward-analysis	Highlighting possible chains of change impact emerging from the initial adaptation of a specific node	Systematic evaluation of possible consequences resulting from impact that propagates by dependency chains
Impact check list	Representation of the nodes that are directly linked to the node in question; classification by active or passive edges is possible	Systematic evaluation of impact propagation in or from the surrounding of specific elements
Mine seeking	Specific application of a feed-forward-analysis; highlighting possible chains of change impact emerging from an adaptation to a user-defined node and affecting previously identified nodes of major structural significance	Systematic evaluation if planned adaptations may result in undesired change impacts to important components
Structural ABC-analysis	Highlighting nodes and edges that are generally involved in characteristic subsets (e.g., feedback loops, hierarchies, clusters, paths)	If the occurrence of a specific structure criterion is to be reduced, nodes and edges that possess potential for its reduction are on hand and can be systematically examined
Trace-back-analysis	Highlighting possible chains of change impact starting from the last element affected back to the probable originators	Systematic evaluation of possible impact originators that affect important system components by impact chains

Mine seeking represents a specific application of a feed-forward-analysis (see definition above). Chains of change impact are pointed out, which reach from nodes that are to be adapted to nodes of major significance in the structure. These destinations of impact chains must already be known as system elements that are unsuitable for adaptations, e.g., because of the unmanageable amount of required change effort. Such “structure mines” generally represent elements possessing a specific structural embedding in the entire structure, such as

busses or articulation nodes. In other words, mine seeking helps developers detect planned adaptations, which would cause indirect impact to system elements that could provoke an enormous adaptation effort if they were affected. Thus, measures of system change, which may seem to be unproblematic, could be identified as harmful because of change propagation effects.

The **structural ABC-analysis** helps identify nodes and edges of a structure that are generally involved in specific structural subsets, such as feedback loops, hierarchies, or clusters. The results of the analysis can be presented in list form that shows nodes or edges and their quantity of appearance in specific structural constellations in descending sequence. Enhanced representations apply portfolios, such as an influence portfolio that comprises the active and passive sums of nodes on the two axes. The knowledge about these elements enables product developers to effectively influence the quantity of these system characteristics. If, for example, a system possesses a large number of feedback loops, system adaptations will often be difficult, because these loops can probably cause self-energizing effects that are difficult to control. Developers may be interested in reducing the number of feedback loops by target-oriented product adaptations. Generally, a feedback loop can be avoided by breaking up one of the dependencies forming the circular path. Several feedback loops can be disconnected by only one structural measure, if they meet in one edge, which can be removed from the structure by product adaptation. Often only a few dependencies are involved in the majority of existing feedback loops; thus, the quantity of feedback loops can be dramatically decreased by eliminating one of these dependencies. An example of a use case for the systematical optimization of a system by eliminating feedback loops is given in Chapter 4.

Contrary to the feed-forward-analysis, the **trace-back-analysis** depicts possible chains of change impacts starting from specifically observed nodes that show the undesired effect; from these nodes dependency paths are traced back to the initial causes. These originators of undesired system changes can be identified by examining the passive dependency chains belonging to the element in question. For each identified dependency of the impact sequence, developers can evaluate whether the consequence observed emerge from this path and can stop further investigations if the path is irrelevant. Thus, analyses are kept to a minimum. Usually, in complex systems not all trace-back routes can be systematically observed and reasonable constraints must be defined. If, for example, product developers know system elements that have not changed in the specific use case (e.g., fuel was identical in two test runs of an engine), dependency chains passing through these elements can not constitute part of the observed change propagation that results in the undesired effect. Therefore, all trace-back routes passing through these elements can be excluded from further observation.

The structure manual can serve as a reference book for product adaptation, i.e., to create variants based on the fundamental structure. As each variant is generated by adaptations to the fundamental product, its structural basis remains constant. For this reason a static implementation of the reference book can be applied. The first part of a structure manual depicts the characterization of a complex structure and its elements in order to provide a better system understanding. Useful methods have been presented in Section 3.5. Additionally, the application of structural knowledge to the practical implementation of engineering changes becomes possible by the methods presented in Table 3-5. These have to

be applied to a pre-selection of sensitive system elements and results have to be introduced to the structure manual. Information about the relevance of these elements for structural consideration originates either from practical system interaction, i.e., the experience of developers, or from the application of basic analysis criteria (see Section 3.5). A disadvantage of a static reference book is that the pre-selection of elements concerned outlines the scope of manageable use cases. Consequently, dynamic software implementation of the structure manual could enhance its practical applicability. Then it would be possible to determine originators or destinations of change impact in user-defined situations and to specifically obtain relevant elements and subsets that have to be more closely considered. Typically, a structure manual is used on demand, i.e., information is looked up for planned or occurring system changes. For the initial creation of the structure manual, experts would be required to specify relevant system elements for consideration by the methods described in Table 3-5.

3.6.2 Structure potentials

Besides the provision of a structure manual for improved interaction with a given structure, discussions of practice can aim at the optimization of a system's structure. In this case developers decide to fundamentally redesign a system's structural composition in order to facilitate future system interaction and adaptation. The identification of optimization measures then helps determine favorable changes to the structure. It must be noted that the discussions of practice considered here are based only on structural information, i.e., resulting propositions of structural changes (e.g., the removal of a specific dependency) must be evaluated successively concerning their possible realization in practical design. Nevertheless, even if some propositions do not seem to be directly applicable, developers become inspired and obtain information about the main challenges of the system from a structural point of view. System elements and dependencies are pointed out, whose adaptation may best increase the relevant system attributes, such as the product robustness in case of change impact. The propositions developed are meant to be used in the context of creativity workshops, where they serve as suggestions for elaborating optimization measures. Three approaches are mentioned as follows: the structural ABC-analysis has already been introduced for application to the structure manual above and can be applied for optimization as well. Tearing represents a DSM-based optimization approach primarily designed for the application of process-oriented networks. Finally, an approach using an evolutionary algorithm is presented that aims at decreasing the structural criticality of an entire system.

Structural ABC-analysis

A structural ABC-analysis provides lists of elements or dependencies, whose structural adaptation results in significant impact to the entire structure in question. Such an analysis can focus on the dependencies that are generally involved in feedback loops. Overlapping feedback loops (see Figure 3-27) can comprise the same elements or dependencies, and every feedback loop can be disconnected by the removal of one of its elements or dependencies. Consequently, the removal of a dependency that is part of a multitude of feedback loops can result in a noticeable decrease of the entire quantity of feedback loops. Of course, the removal of a dependency has to be feasible in the technical design of the product. As this can not be

assured for every dependency in every practical application, the structural ABC-analysis provides a list of dependencies that represents candidates for removal. The dependencies are sorted by the quantity of their associated feedback loops and can be sequentially evaluated concerning possibilities of adaptation. The application of a structural ABC-analysis on the reduction of feedback loops is represented in Section 4.1.1 as an example of a case study. Further structural ABC-analyses can, for example, help determine the most suitable dependencies for the reduction of hierarchical subsets (avalanche effects).

Tearing approach

Tearing represents an optimization approach by means of the analysis methods presented. It is based on the selective elimination of dependencies that are identified by matrix realignment. Therefore, tearing requires the application of a partitioning algorithm (see Section 3.5.3), which tries to accumulate dependencies at one side of the matrix diagonal. If this alignment can not be realized completely, partitioning tries to arrange dependencies as close as possible to the diagonal. In time-based structures, resulting dependencies that are located below and far away from the matrix diagonal represent the causes for long iteration chains. The tearing method selects such dependencies and proposes their elimination in order to optimize the structure. The dependencies selected by tearing depend on the quality of the preceding application of partitioning. Often several different matrix alignments can result from partitioning algorithms if applied to the same structure. As the selection of dependencies for the tearing optimization results from the matrix alignment, it is difficult to guarantee an appropriate selection of dependencies that are proposed for elimination. In Chapter 4 the consequences of this disadvantage of tearing are shown in a use case example. Here, the application of the tearing method results in less feedback loop reduction than can be obtained from another optimization approach. Altogether, tearing can not be recommended for optimizing complex structures, as the selection of dependencies as an optimization objective is based on a partly randomly ordered matrix alignment. Nevertheless, an explication of the tearing approach is included in the Appendix.

Evolutionary algorithm for reduction of system complexity

A promising possibility for the optimization of system structures represents the application of an evolutionary algorithm (EA). If it is possible to establish an authentic index for comparing the complexity of different structures, a powerful and automated optimization procedure can be implemented that identifies required structural changes.

Reasons for applying an evolutionary algorithm

Theoretically, the ability to analyze any given structure allows for its optimization. If dependencies are varied systematically, resulting structures have to be analyzed, and the best one can be selected. Actually, this represents the only method that guarantees finding the optimum configuration of a system structure. Of course, this procedure is not only inefficient but impracticable in practice; even in systems that consist of only a few elements and dependencies the number of possible configurations is immense. A system consisting of n elements and a variable number of dependencies between them that can adopt m different weight specifications possesses $(m+1)^{n^2-n}$ possible structural configurations. That means

four possible system configurations for a system comprising two elements and only one non-weighted dependency type; however, a system with 25 elements and two different dependency types (representing, for example, low and high change impact between components) can theoretically adopt 3^{600} different configurations. The challenge of structural optimization is to find high quality results in adequate time, instead of a complete permutation of all possible constellations.

General structure of an evolutionary algorithm

Evolutionary algorithms (EAs) adapt the mechanisms of evolution and genetics in order to optimize complex systems. Two models that have turned out to be appropriate for computer simulation and IT applications are the evolutionary strategies (ESs) [SCHWEFEL 1995] and the genetic algorithms (GAs) [GOLDBERG 1989]. Both were developed simultaneously but independently, and their differences are not relevant for the approach considered here. In the recent literature, elements of both strategies are combined and terms are equally used [MIETTINEN 1999]. In the following, the steps of a common EA are listed, and the specific application for structural optimization is presented.

- **Coding:** Information on the system's structure needs to be encoded to enable the processing by a computer; here, vectors of real numbers are applied. These vectors are called chromosomes, according to natural evolution and genetics. The components of the vector represent the structure's dependencies and are called genes; their values, which represent the dependencies' weights, are called alleles. An initial population of chromosomes can be generated and rated afterwards. Two different parent chromosomes are necessary: the first one represents the given structure; the second one is generated by reproducing and mutating the first chromosome.
- **Reproduction:** Two offspring chromosomes are generated by copying the parental chromosomes.
- **Recombination:** The two offspring chromosomes are recombined, which means the alleles of each gene are switched according to a certain probability.
- **Mutation:** One gene of each offspring chromosome is chosen and its allele is varied.
- **Rating:** The chromosomes are rated to enable the successive selection.
- **Selection:** The two best chromosomes are chosen to be the next generation parent chromosomes. A generation is represented by an iteration of reproduction, recombination, mutation, and rating. The total number of generations is preset by the user. If the abort criterion, the maximum number of iterations, is reached, the current best chromosome represents the result.

Possible metrics for structural optimization

Parameters of the EA are varied by recombination and mutation in order to optimize the system considered. Here the parameters are the dependencies (edges) between the elements (nodes). Possible structural variations are removing edges, reducing edge weights, and adding

edges. Generally, raising weights of edges is also possible, but does not improve a structure's complexity.

Next, the question of if and why these variations lead to structural improvements must be answered, which depends on the meaning of applied dependencies and weight meanings. Removing dependencies can lead to improvements of the structural complexity, as mutual impacts of elements are reduced. Also adding edges may lead to positive effects on a system's complexity, depending on the meaning of dependencies. If they represent the flow of information in a communication network, adding dependencies may improve communication, since it raises connectivity and leads to shorter alternative connections between communication participants. If dependencies represent the change impacts in a product architecture, adding dependencies means adding development constraints and therefore results in a negative effect, as it complicates the structure.

Besides the meaning of the dependencies, the significance of dependency weights also has to be taken into account. If dependency weights represent lengths (e.g., of cables in an electric product), all structural variations mentioned can potentially improve the structure considered; removing edges always reduces the number of dependencies and thereby the amount of complexity. Reducing dependency weights shortens distances. In contrast, raising dependency weights lengthens distances or reduces bandwidths. Adding edges increases the number of dependencies in a system, but may nevertheless improve the structure, because additional edges can result in alternate connectivity between specific nodes. That means that such subsets contain more edges than the original ones, but may possess shorter paths between elements. This scenario is clarified by the variation of the subset depicted in Figure 3-42. The original subset at the left side can be improved by adding an edge between node 3 and node 2, as is shown at the right side of the figure. The resulting structure contains one more edge, but possesses the shorter (indirect) path between the nodes 1 and 2 than the initial structure.

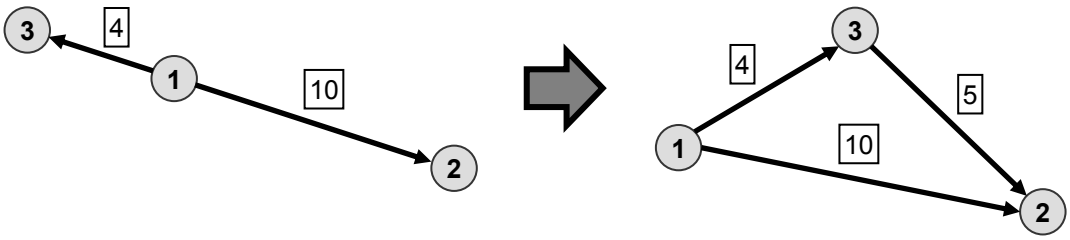


Figure 3-42 Optimizing a structure by adding an edge

It is not possible to properly display the lengths in the strength-based graph (see also Figure 3-29 and Figure 3-30); in Figure 3-42 the indirect path from element 1 to element 2 that passes to element 3 is longer than the direct connection from element 1 to element 2; however the belonging weights show that the indirect connection is shorter (9) than the direct one (10). A practical application of adding dependencies could be to implement additional connections for improving the information flow by a faster transfer of information (shorter distances mean faster transfer). This can increase the efficiency of a communication network, although information must pass by more intermediate elements.

If the meanings of dependency weights represent change expenditures, only removing edges and eventually adding edges will lead to structural improvement. Change expenditure describes the required effort of adaptations to system components due to an initial change in one specific component. Here the optimization target is to minimize the required effort of adaptations. Removing edges always reduces the number of dependencies and thus the amount of complexity. If a dependency between two components can be avoided, the second component does not require any adaptation due to any changes implemented to the first one. In contrast to the meaning of lengths previously mentioned, adaptations to the level of change expenditures are impossible; these are determined by the system environment which cannot be adapted by measures to the system. If, for example, an expensive tool is used to produce a product component, changes to this component may require expensive changes to the tool. The only way to reduce these expenses is to design a substitute component with less production cost. That means that the original component (the original node and its dependencies in the structure) has to be replaced by a newly developed component (the new node and its dependencies). However, the exchange of system elements is not taken into consideration for the structural optimization measures mentioned here. For this reason, if dependency weights represent changing expenditures, only dependencies and not their weights can be objects for a structural optimization.

Fitness function for rating structural constellations

The measures of system optimization presented are applied to the mutation step of the EA. Mutated structural constellations must then be rated in order to determine probable ameliorations. In industrial practice, the decision to change a system's structure depends on the required expenses as well as the attainable improvement. This has to be considered during the conception of the rating algorithm. Apart from that, structural complexity depends on the number of dependencies a system contains. Generally, the fewer dependencies that exist, the less complex a system is. Consequently, a completely unconnected system represents the least complex one. Obviously, this can not be a desired alternative, as completely disconnected elements impede the functionality of composed assemblies. A completely unconnected communication network, for example, is the least complex, but does not enable communication at all.

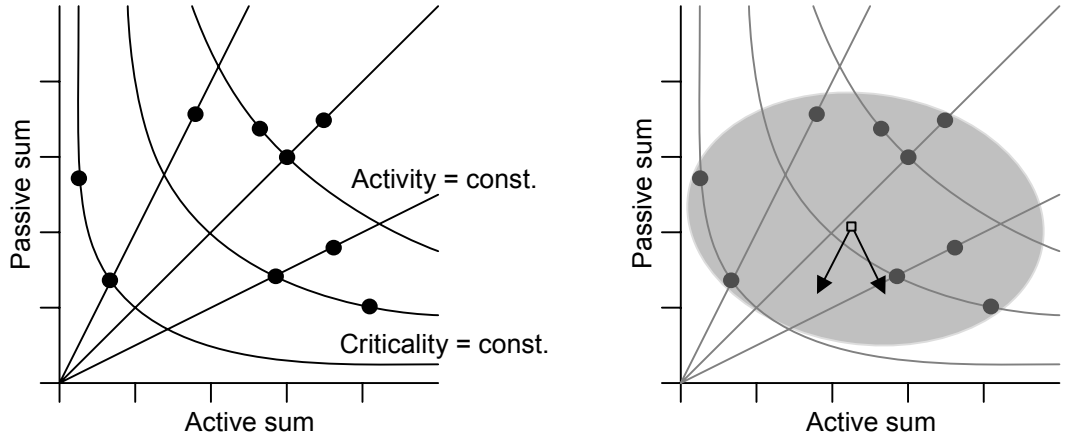


Figure 3-43 Reduction of system criticality as an optimization criterion

The influence portfolio makes it possible to rate the criticality of single nodes of a structure by their active and passive sum (see Section 3.5.3 and the Appendix). Generally, the closer a node is located to the lower left corner of the portfolio, the less critical it is, as its interconnectivity with other nodes decreases. The rating of nodes can be enhanced for the entire system, if the mean value of all nodes' active and passive sum is computed. If measures of structure variation result in a decrease of this mean value, the system's criticality decreases as well, as illustrated in Figure 3-43. Such a decrease of criticality can be interpreted as the simultaneous decrease of system complexity, as simplified dependency constellations can be better managed.

Unfortunately, this index of overall system complexity does not take into account all relevant aspects, as shown in Figure 3-44. Starting from the initial structure at the top of the figure, both structures below are derived by removing one single dependency. In the influence portfolio, this would result in an identical decrease of complexity, as removing one edge means the loss of exactly one active and one passive dependency value for the entire structure. Consequently, every edge that is removed from the structure results in the same decrease of complexity, even if derived structures are highly different in their internal connectivity. Continuous measures for decreasing complexity would lead to a complete disconnection of the structure and are not a desired solution, as explained above. Consequently, an additional criterion is needed to avoid the algorithm converging towards a trivial solution.

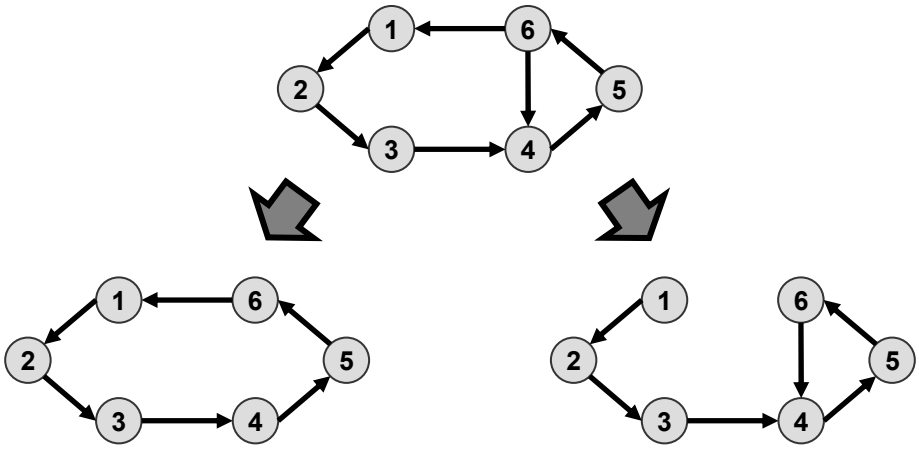


Figure 3-44 Reducing system complexity by removing edges

The approach presented here applies a structure index that is based upon FREEMAN's work on centrality [FREEMAN 1979]. The basic idea is that structural characteristics can be expressed by the analysis criteria attainability and closeness, which are considered in Section 3.5.3 and are mentioned in the Appendix. Removing an edge could decrease the attainability but simultaneously increase the closeness of a specific node in the structure. It could also decrease one node's attainability and decrease another one's closeness; any combinatory effect is conceivable. If users can specify the desired evolution of attainability and closeness for a structure in question, a combined index comprising these criteria for all nodes in a system can be applied for rating the system complexity. This can be explained by again using

the structural adaptations in Figure 3-44. In both cases one dependency is removed from the original structure, but the different measures result in different values of attainability and closeness; in the left structure the attainability still remains the same compared to the original structure (as all nodes can reach each other by a dependency path). In contrast, in the right structure, the elements 1, 2, and 3 can no longer be reached by the other ones—resulting in a significant decrease of overall attainability and closeness.

In the following Equations 3-15 to 3-19, the computational scheme for the determination of the fitness value by application of the attainability and closeness of nodes is shown.

$$f_{active} = v_a \cdot Attainability_{active} + v_b \cdot Closeness_{active} \qquad \textit{Equation 3-15}$$

$$f_{passive} = v_a \cdot Attainability_{passive} + v_b \cdot Closeness_{passive} \qquad \textit{Equation 3-16}$$

$$F_{active,arithm} = \frac{1}{n}\sum_{i=1}^{n} f_{active,i} \qquad \textit{Equation 3-17}$$

$$F_{passive,arithm} = \frac{1}{n}\sum_{i=1}^{n} f_{passive,i} \qquad \textit{Equation 3-18}$$

$$F_{total} = w_a \cdot F_{active,arithm} + w_b \cdot F_{passive,arithm} \qquad \textit{Equation 3-19}$$

The single fitness values of each node are composed by means of the structure criteria attainability and closeness, as can be seen in Equation 3-15 and Equation 3-16 (definitions for attainability and closeness are available in Section 3.5.3 and the Appendix). Thus, active as well as passive attainability and closeness values must be determined for every node of a structure. The variables v_a and v_b represent weighting factors that allow users to mutually specify the desired ratio of attainability and closeness, depending on the desired outcome of the optimization procedure.

Next, the active and passive fitness values of all nodes have to be integrated with averaged values. These are typically computed by the arithmetic mean of the single fitness values f_{active} and $f_{passive}$ of every node in a structure (Equation 3-17 and Equation 3-18). Alternatively, the geometric mean or a peak value can also be used for computing the total fitness value.

Finally, the total fitness value F_{total} is composed of the averaged active and passive fitness values $F_{active,arithm}$ and $F_{passive,arithm}$. The variables w_a and w_b represent weighting factors that allow users to mutually specify the desired ratio of activity and passivity fitness according to the specific requirements.

Whether the attainability and closeness criteria will be minimized or maximized depends on the specific application. For example, communication networks and product architectures can represent conflicting examples. In communication networks a high attainability is advantageous, because this means that many system participants can communicate with each other. In contrast, a low attainability is advantageous in product architectures, where dependencies represent change impact between components. In this case only a few elements are affected by implemented changes to other elements. If attainability and closeness values of all nodes of a system are combined, the resulting index is appropriate for rating the

structural complexity. Based on this a genetic algorithm can be implemented. An example of an application is shown in Section 4.1.

3.7 Requirements on the software implementation

As already mentioned in the preceding sections, software support is required for an effective application of the approach presented on structure-based complexity management. General requirements of the layout of an implementation are classified by the main steps of the approach and are represented in Figure 3-45. Below, the different modules are described. Details about certain aspects are additionally discussed by MAURER ET AL. ([MAURER ET AL. 2004], [MAURER ET AL. 2005]).

System definition

The initial task of defining the system scheme can only partly be facilitated by software support. The specific problem has to be transferred to a structural description form for subsequent investigations. Users have to identify and systematically prepare the required domains and dependency meanings connecting them. Therefore, the application of a conceptual MDM form is recommended. A generic template is presented in Figure 3-6, and Figure 4-7 shows a form filled with data from a case study example. Even if the general MDM layout does not demand software support, capturing this information represents the fundamental input for the data storage, which is accessed in all following process steps as well. The concept of the MDM form is the basis for subsequent information acquisition, where selected sub-matrices are filled with dependencies.

Information acquisition

To support the acquisition of dependency information, a software implementation has to provide a DSM as well as a DMM form for specifying the dependencies of MDM subsets. Several features are conceivable that can facilitate the acquisition, especially if large structures have to be processed. If it is possible not only to sequentially place elements along the column and row heads of a matrix but to hierarchically classify them, information acquisition can be improved. For example, the elements “engine” and “seat” can be created on the highest hierarchical level, each with a multitude of subordinated components. If during information acquisition, it can be determined that both elements are independent from each other, dependencies between subordinated components do not have to be considered any more. Therefore, systematic hierarchical classification of elements considered can significantly reduce the effort required for information acquisition.

In addition to the implementation of dependencies, it also has to be possible to flag each matrix cell after its consideration. This could support the matrix filling when it is done in separate workshops or when processing large matrices, as users can clearly see those matrix cells that remain for consideration. If, in contrast, only the implementation of dependencies is provided, users can not differentiate between a deliberately non-established dependency and a matrix cell that still remains for consideration.

Furthermore, it would be helpful if besides the dependency actually considered, the inverse one could also be specified at the same time. In many applications this allows for faster

acquisition, as users can declare the status of both possible dependencies between the same two elements in combination. Generally, the matrix form is only required for providing an orientation in the structure during the acquisition process; in fact, users do not need to extract structure elements from column and row heads in order to discuss their probable dependencies (as is required when applying common matrix forms). Software support can facilitate the acquisition process by not representing the entire matrix but only the two elements actually discussed (eventually enhanced by visualizing pictures or schemes of the elements). Thus, workshop participants can concentrate on the dependencies in question, and the matrix is filled in the background.

Documentation is required during the information acquisition to save details on the process of decision-making concerning established (and consciously non-established) dependencies. If users discuss the existence of a dependency during information acquisition, it can be assumed that later structure considerations will again lead to different opinions about these exact dependencies. The documentation of decision-making must be supported by software due to the number of possible dependencies and the need for fast and easy access to the documentation. Experts that are required for the acquisition workshops are typically only available for a limited time; thus, statements and explications are best stored directly as attachments to the decisions made (e.g., by attaching the documentation to the edge by a hyperlink). In addition, the documentation of non-established edges must be possible as well. In this context, an advantage of matrices for information acquisition is obvious, as non-existing edges are difficult to represent in graph depictions and therefore comments can not be attached to them. In matrix forms, a matrix cell exists for every possible edge, independent from its existence. Furthermore, decisions about both edges and nodes must be documented, to clarify, for example, the interpretation of their meaning.

As during the acquisition of structural information, the status of knowledge about the system continuously increases, possibilities for plausibility checks on the captured data have to be enabled by appropriate software support. For example, the quantity of indirect dependencies between two nodes can be determined by algorithms (see Section 3.5.3) and often helps determine the existing direct dependencies between two specific nodes [EICHINGER ET AL. 2006]. Visual highlighting of specific matrix cells can help users focus on relevant aspects for iterative examination.

Besides information acquisition by interviews, structural connectivity can be determined by use of existing data sets from other modeling tools. Therefore, an appropriate software implementation has to provide import interfaces. Structural information can emerge from a variety of software applications applied to product development: Product Data Management (PDM) software handles large amounts of data that occur during the product life-cycle. Such applications allow access to product-related data via a predefined product structure[36]. Product configurators map existing product topologies and create rules that express possible component combinations and mutual exclusiveness. With these systems, the compatibility of components, for example, can be defined and even linkages from components to requirements or functions can be implemented. Tools supporting the aspect of variant management can be

[36] Typically, a hierarchical breakdown of the overall product into geometrical subsets is predefined.

seen as specific applications of product configurators, as product variants are specified from a comprehensive product program [SCHUH & SCHWENK 2001, P. 245FF]. These tools support the improvement of existing modularization approaches from a component-oriented point of view. Attention is especially paid to the consideration of quantities sold and the benefit from product units. Project management tools are designed for establishing, managing, and optimizing process flows, whereas related documents can, for example, be linked to specific process steps. For example, Gantt charts depict optimized sequences of process steps by minimization of iterations and identification of tasks that can be executed in parallel. Process management tools support the design, analysis, and optimization of flow-oriented networks, such as information flows or logistic processes. Process networks often include hundreds or thousands of elements. Typical network analyses executed by support of process management tools are, for example, the identification of bottlenecks, critical paths (possessing minimal time buffers), or undesired iteration loops. Tools for the visual mapping provide intuitive possibilities of network creation and visualization that are even appropriate for creativity-based team meetings. Often, standard office applications are applied that are enhanced by specific macro functions. The functionalities of modern CAD systems, especially parametric design and feature techniques provide, for example, structural information about geometric change propagation. If dependencies that are already established can be implemented from these different applications for a software-based MDM approach, information acquisition can be efficiently realized in many use cases.

Modeling

In the process step of structure modeling, the visualization of networks and the access for users have to be enabled by software support. According to the requirements and explications on modeling in Section 3.4, representation forms for DSMs, DMMs, and strength-based graphs have to be provided. As results from structural analysis and optimization have to be mediated for users by appropriate structure modeling, the bi-directional connectivity between representation and computation is required; i.e., analysis results have to be accessible the same way as the original structure. Matrix forms are realized in common spreadsheet applications and specific tools for graph representations are available as well; however, the simultaneous use of both depictions for visualization and structure interactions requires its own application that integrates both forms in a combined approach. Besides access to the central data storage, the step of modeling requires additional access to filters and analyses, such as for representing subsets of a structure. Filters serve to extract parts of the network in order to allow focused considerations. In addition, the application of algorithms enables users to generate and represent structural constellations that are based on the existing structures, but have not been explicitly on hand. In general, the provision of suitable possibilities for representing structural content is necessary for the process steps of information acquisition, structure analysis, and discussion of practice. This is depicted by the vertical arrows in Figure 3-45.

A further objective of system modeling is to support communication processes of development teams and to facilitate the understanding of discipline-spanning connectivity and its effects. Therefore, the modeling approach must be applicable for investigations of the structure during team meetings. That requires that modeling functionalities be executed with

standard hardware and software, i.e., through the use of notebook or desktop computers. As the visualization in team meetings is generally realized by video projectors that are limited in screen resolution, the modeling approach must be sufficiently plain in order to represent comprehensive content within the available display limits.

Structure analysis

Structure analyses require the effective application of filters and algorithms to structural content. For this reason, a compilation of basic filter and analysis components is recommended, which can be combined by users by means of an analysis wizard. Only the provision of flexible structural interaction allows for the effective application of analysis on user-defined structure contents. For example, the isolated implementation of a feedback loop filter does not meet the analysis requirements of typical use cases (e.g., because of too many resulting loops). Only the possibility of combining this filter with further ones allows for target-oriented use. An example could be a feedback loop analysis in combination with a specifically selected node that has to be included in the resulting loops. Also, the maximum length of feedback loops could help restrict analysis results to case-specific needs. The multitude of combinatory possibilities of the basic analysis criteria (see Section 3.5 and the Appendix) asks not only for the modular design of filter components, but also for a graphical representation of the components' compilation of comprehensive filters.

Discussion of practices

Software support has to enable users to specify exactly the purpose of structural considerations and requirements for the solution before creating a structure manual or propositions for structural improvement. Generally, it would be possible to display a fixed selection of required analyses in the form of a structure report; however, particularly for comprehensive structures, this would cause enormous amounts of information that could hardly be managed and interpreted by users. Additionally, it could not even be guaranteed that the output would meet the initial requirements. Furthermore, a software-guided definition of analysis and optimization target possibilities for setting up a compilation of required filter and analysis views are required. These can be used for creating the structure report, which can then be used as a reference book for structural adaptation. The determination of possibilities for the structural optimization also requires efficient software implementation in order to allow the application of comprehensive network data.

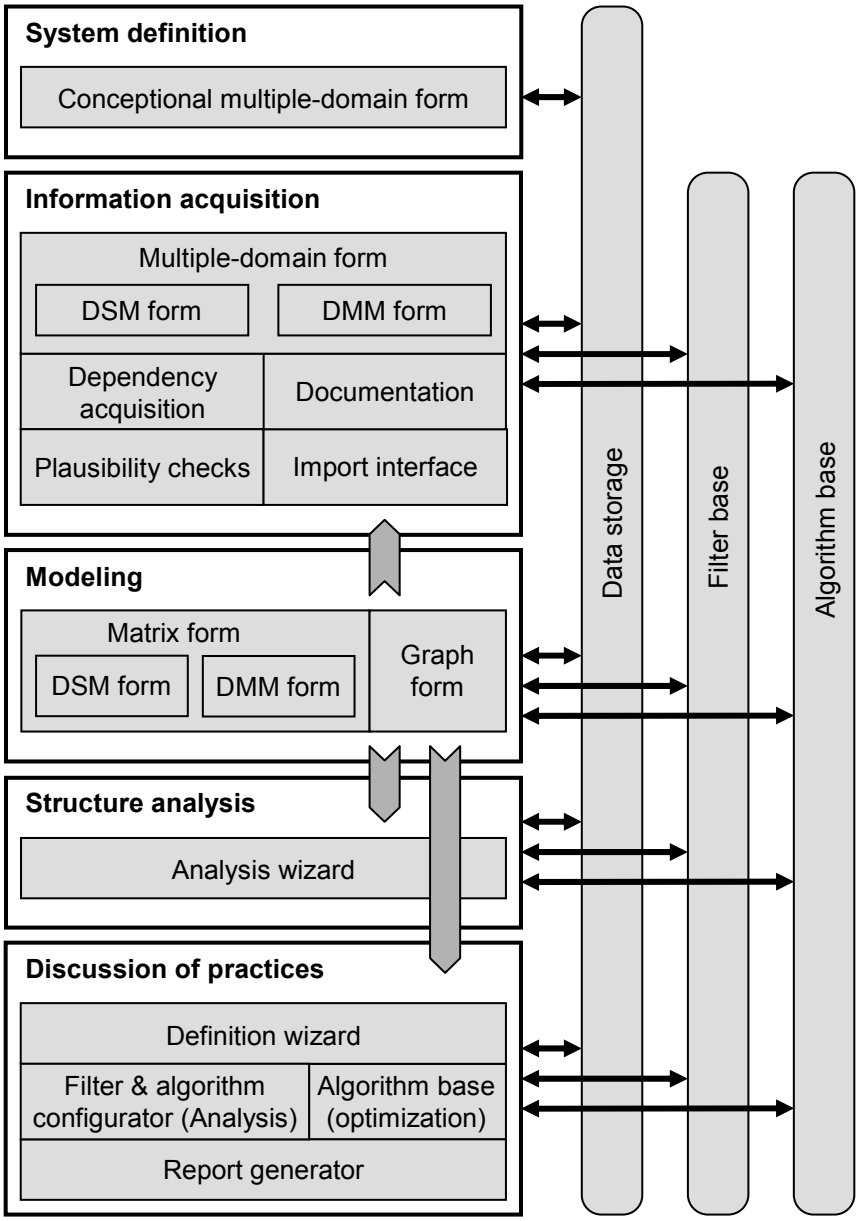


Figure 3-45 Requirements on a software implementation supporting the structure management approach

4. Verification

In the present chapter, the structure management approach is applied to selected use cases in order to clarify its practical application and to verify obtainable benefits. First, two examples of DSM structures are considered by methods developed in this thesis in order to gain a better system understanding, to predict system behavior, and to propose possibilities of optimization. A further example represents an entire MDM approach and illustrates the computation of relevant DSMs from available sub-matrices. These derived structures are then applied to DSM analysis and interpretation.

4.1 Analysis by use of the Dependency Structure Matrix

The initial case study is taken from an industrial research project and considers the analysis and possibilities of optimization of the structural design of a packing machine. Four different approaches are presented that apply methods presented in this thesis and whose results can be compared. A second case study presents a DSM structure of an automotive engine. Emerging from an industrial research project, this structure presents additional methodical analyses. Whereas the first case study focuses on the structural criterion of feedback loops and their best possible reduction, the second case study consists of too many feedback loops for an efficient analysis. However, alternative analyses are executed on this structure that may significantly improve understanding of the system.

4.1.1 Structure analysis and optimization of a packing machine

In the following the use case, an industrial packing machine will be closely examined. The initial problem of the manufacturer arises from frequent product adaptations that often cause unexpected side effects and require more development resources than initially planned. The original cause for these adaptations arises from a multitude of individual customer demands. That means that it is extremely difficult for the manufacturer to reduce the degree of adaptive design without compromising customer choice. The main source of income is the production of customized machines, as standardized machines are rarely sold. For this reason, a simplification of the product development can not be realized by a reduced customer-oriented design, as this represents the majority of orders. The management identified the lack of system understanding concerning short-term product adaptations as the major internal problem of the company, which has led to unprofitable development projects.

Objectives

The first objective was to obtain a greater awareness of the unknown interrelations between product components by applying a structure analysis to the packing machine. Furthermore, the company management wanted to minimize the generation of product variants as much as possible. Opportunities and restrictions for product adaptations must also be systematically identified in order to facilitate the daily work of product developers. In the case of a specific

customer request, developers should be able to rate the feasibility and required measures and resources quickly. Altogether, a structure analysis should provide a better understanding of product dependencies; suitable optimization measures should enhance the product robustness due to component adaptations. As new and unscheduled design adaptations must be implemented in the future, developers also need to know as early as possible about the consequences of such changes. In an ideal situation, one specific change measure only affects a few parts of the entire product, i.e., an optimized product structure allows more flexibility in individual design.

The product components mentioned in the following structure examinations are anonymous due to a nondisclosure agreement. Nevertheless, the analysis, conclusions, and propositions of optimization can be clearly retraced and allow for transfer to other component-based analyses. The structure is comprised of the 25 main components of the packing machine. Dependencies between the components represent possible change impact, i.e., the adaptation to one component may result in a change impact to other components connected to the first one by dependencies.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
A		X									X		X							X			X		X
B	X												X							X				X	X
C	X			X		X				X															
D	X	X	X							X															
E			X	X		X										X			X		X	X			
F																									
G	X							X			X	X			X	X									X
H	X						X					X			X	X									X
I	X			X							X														
J	X	X	X	X					X															X	
K	X	X							X			X	X	X						X	X				
L	X							X			X				X	X									
M	X	X																		X					
N											X	X													
O	X						X	X				X				X									
P								X																	
Q							X																		
R																X	X								
S	X	X				X			X													X			
T	X	X											X												
U	X		X	X	X	X	X			X	X		X	X	X	X	X	X	X				X	X	
V	X																								
W	X																							X	
X																							X		
Y	X	X																							

Figure 4-1 DSM representing the main modules of a packing machine

The dependency network can be seen in Figure 4-1 in a matrix depiction, and it was acquired through interviews with product developers. The initial task in structure analysis was to identify the problem mentioned in the structure, i.e., unexpected side effects due to product adaptations had to be assigned to structural characteristics of the structure in question. If this was not possible, the problem probably was not caused by structural inadequacy, but emerged from other, non-structure-related failures. If, for example, the manufacturer missed an important technology trend, structural optimization would not help solve the problem.

Determination of structural characteristics

First the partitioning analysis is applied to the structure, i.e., the matrix rows and columns are realigned in order to bring as many dependencies into the upper triangle of the matrix and as close as possible to the matrix diagonal. As described in Section 3.5.3, this procedure indicates highly linked element groups as well as the elements' tendency towards active or passive behavior. The partitioning could be done manually or by a software macro that applies step-by-step instructions. As the entire structure comprises only a few elements, and the degree of connectivity is rather low, the realignment does not ask for major computational efforts. The resulting matrix alignment can be seen in Figure 4-2. It has to be understood that this alignment does not represent the only solution; in fact, several other alignments are possible, as the optimized positioning of one dependency in the matrix can downgrade the alignment of another one. The results of the partitioning analysis show three clusters comprising four to six elements each (O-G-H-L, J-C-I-D, Y-W-T-M-B-A). Furthermore, the element U represents an active, and the element A a passive, bus. Many other dependencies exist outside of the clusters and even link between them. This can be taken as a first sign of the high quantity of existing feedback loops, as dependency chains are spread over almost all elements.

The result of partitioning seems to permit applying the tearing method. This can help improve the degree of independence between the three clusters and advance the modularized character of the product structure. In particular, the designated dependencies 1 and 2 are likely candidates to be removed by tearing. If they could be eliminated, the cluster alignment could be optimized and iterations, i.e., the mutual change impact within a group of components, could be reduced. However, as already mentioned, the matrix alignment by partitioning does not represent the only significant alignment. It is important to notice that dependencies 1 and 2 are only exposed in the matrix due to the selected alignment. If the partitioning were to be executed incorrectly or with a low quality, the subsequently selected dependencies for tearing could be totally different and perhaps even counterproductive for structural optimization.

As mentioned before, the structure considerations were primarily initiated because of the unexpected change effects that occurred, i.e., the unknown change propagations caused by product adaptation. This system characteristic does not come with the identification and optimization of a module design as the main target. Modules concentrate dependencies internally and try to standardize or minimize external links. These modules can then be exchanged without severe adaptations to the entire system. However, in the present use case future adaptation requests can not estimated beforehand. If a part of a module were to be changed, many impacts would result to all further module components. A number of

subsequent adaptations or the exchange of the entire module would be the consequence. That means that modularization can only be effectively applied, if points of future product adaptation are known and the modules are designed to these requirements.

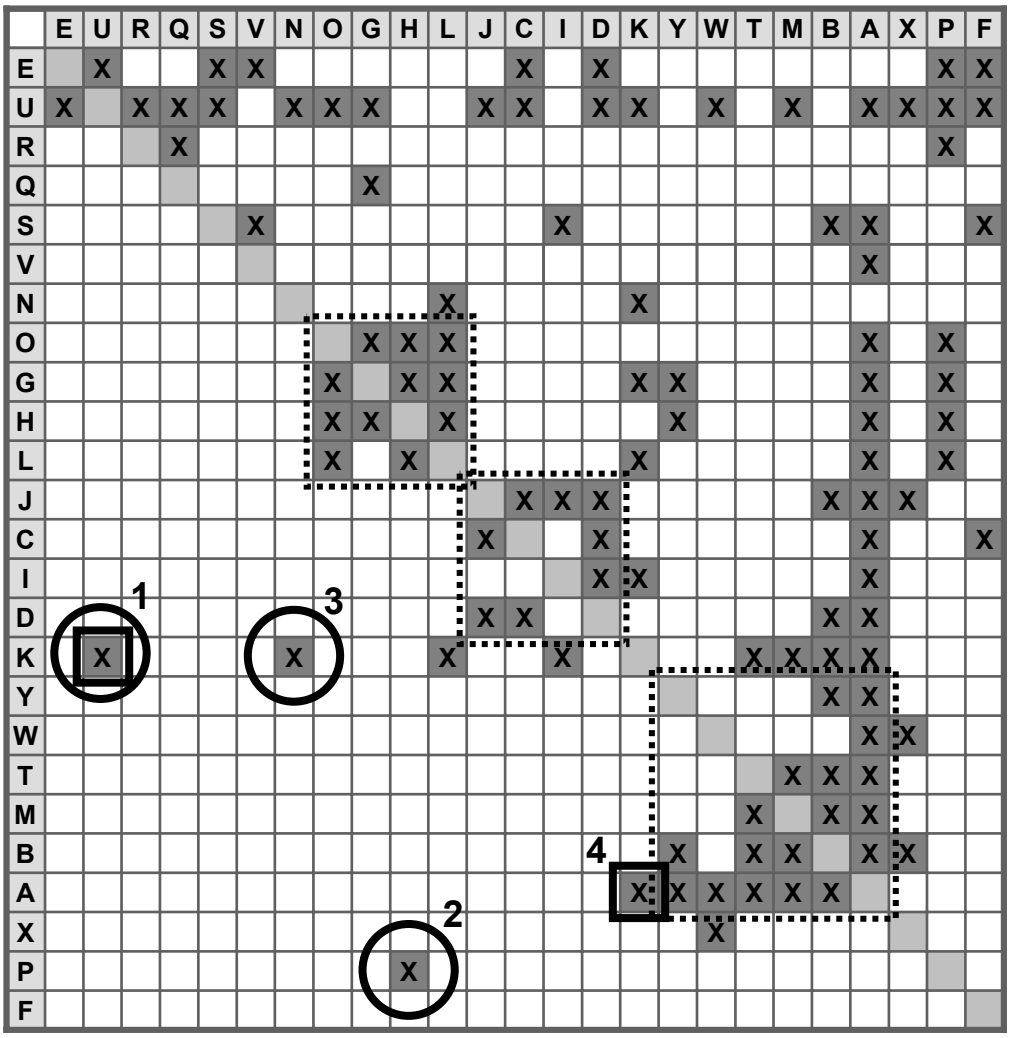


Figure 4-2 Triangularized DSM of a packing machine

In contrast, the situation described before requires a general decrease of change propagation, because changes due to customer request can not be narrowed down to specific product components. For this reason, it is necessary to have a closer look at existing feedback loops, as these may cause iterating change propagation with considerable impact to many components of the entire system. The analysis was executed by a software macro using common algorithms from graph theory and resulted in 1388 feedback loops. This high quantity was not expected initially and allowed further specification of the optimization target. Propositions must be provided for minimizing the amount of feedback loops by as little elimination of dependencies as possible. If these can be realized, newly emerging adaptation requests, even if they are unknown, will no longer provoke large quantities of unmanageable circular change propagations.

Approach 1: Feedback loop reduction by tearing

As two dependencies have already been identified by the previous tearing method (dependencies indicated by 1 and 2 in Figure 4-2), these will be examined for possible reduction of feedback loops in case they are eliminated from the structure. If it is assumed that both dependencies can be avoided by implementing appropriate technical development measures (e.g., the implementation of standardized interfaces), 250 feedback loops would remain in the optimized structure. If this number of feedback loops is still too large, the tearing approach would indicate dependency 3 as the most appropriate candidate for further elimination. If this dependency can also be avoided, only 184 feedback loops remain, corresponding to a percentage of approximately 85% of eliminated feedback loops by only three adaptations to the structural design.

Approach 2: Feedback loop reduction by structural ABC-analysis

An even more effective possibility for the reduction of feedback loops examines the dependencies which are most involved in the existing feedback loops. A specifically designed algorithm passes through all existing feedback loops in the structure and sums up the number of appearances of each dependency in the loops. The application of this algorithm points out the dependencies assigned with the numbers 1 and 4 in Figure 4-2. Both these dependencies constitute part of the majority of existing feedback loops, i.e., the dependency chains of most feedback loops pass through these two dependencies. Dependency 1 occurs in 1200, dependency 2 in 1014 of all 1388 feedback loops. Assuming that both dependencies can be eliminated by appropriate technical adaptations, only 94 feedback loops would remain in the structure. This means a reduction of more than 93% of the feedback loops by avoidance of only two dependencies and is even more effective than the tearing approach presented. Dependency 1 was considered by both approaches; dependency 4 would have not been identified as a relevant dependency for the tearing procedure. In Figure 4-2 dependency 4 is closely aligned to a clustered subset and would therefore not be considered as an object for elimination by tearing. The alignment of the matrix does not suggest that far-reaching iterations pass by dependency 4. The advantage of the systematic analysis of feedback loops compared with the combined partitioning and tearing approach is to better determine the procedure that clearly identifies the most involved dependencies. The limits of applicability for counting the dependencies' occurrence in feedback loops constitute the computational efforts that are required for the determination and sequential processing of all feedback loops in a system structure. Systems comprised of only 30 elements can contain extremely large quantities of feedback loops that are difficult to compute. If, however, the analysis can be computed, but many dependencies are equally involved in a multitude of feedback loops (as may occur in intensely linked structures), the determination of distinct optimization measures becomes difficult.

Approach 3: Feedback loop reduction by combined basic analysis criteria

Another method to determine the dependencies relevant for feedback loop reduction does not even require algorithmic support, but is based on appropriate visualization and logic analysis. As mentioned before, "Element U" represents an active bus in the structure, whereas

"Element A" represents a passive bus. That means that many dependencies are emitted by "Element U", and "Element A" collects many dependencies. Thus, it seems to be obvious that a multitude of dependency paths exist from "Element U" to "Element A". Consequently, many feedback loops will also be composed of outgoing dependencies of "Element U" and incoming dependencies of "Element A". In order to close the feedback loops, the dependency paths reaching back from "Element A" (the collector) to "Element U" (the emitter) are important. The significance of the constellation becomes clear through the representation of the relevant structural subset in Figure 4-3. It depicts the elements that are connected from element U by active dependencies and the elements that are connected to element A by passive dependencies. That means the figure is composed of the active surroundings of "Element U" and the passive surroundings of "Element A".

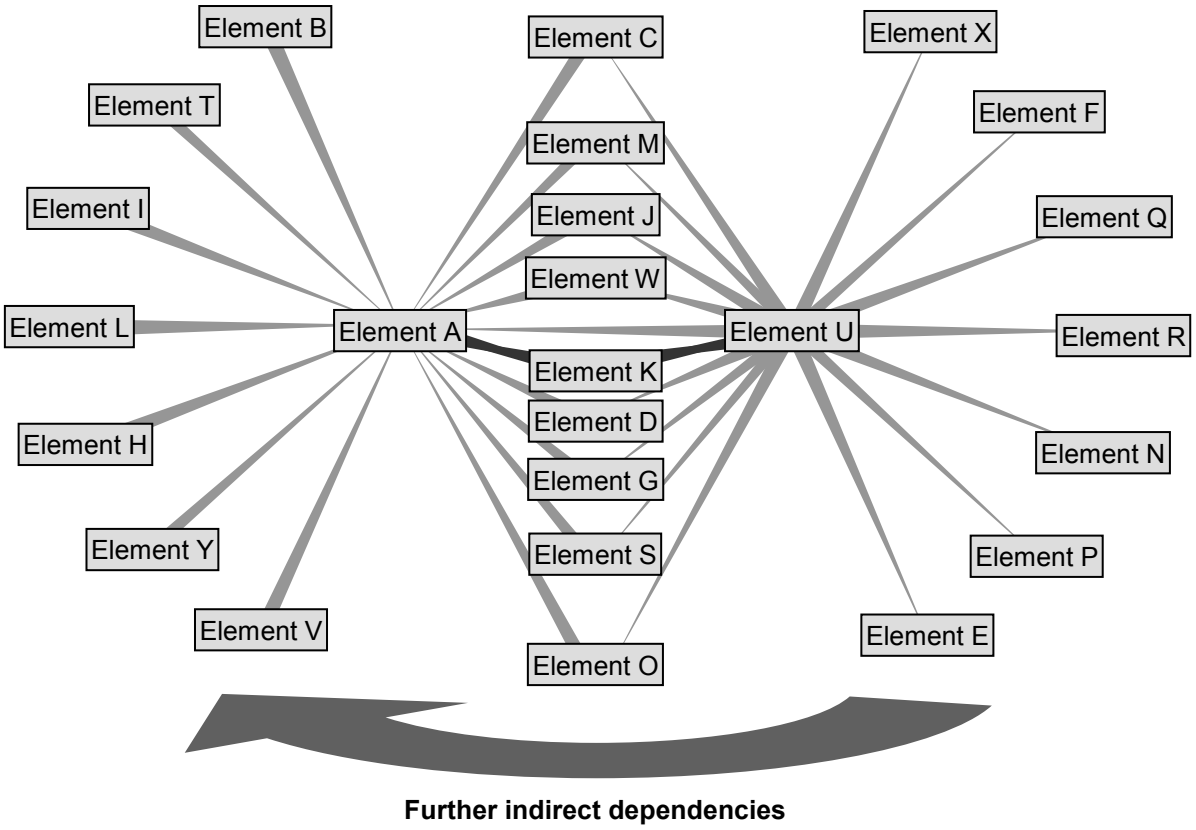


Figure 4-3 Subset comprising the direct surrounding of the active and passive bus elements

Additionally, the shortest dependency path from "Element A" (the collector) to "Element U" (the emitter) is depicted (the dark grey arrows represent bi-directional dependencies). The shortest connection from A to U passes through "Element K" (A→K and K→U) and is the only path reaching from the "collector" to the "emitter" by only one intermediate element. Feedback loops comprising outgoing dependencies of "Element U" and incoming dependencies of "Element A" will pass with high probability through the identified path in order to close the loop. This is proved by the fact that the two path-building dependencies (A→K and K→U) comply exactly with the two dependencies identified by the feedback loop

analysis executed before. Obviously, if the connection between the main receiver of dependencies and the main emitter of dependencies is broken up (i.e., corresponds to the elimination of these dependencies), the majority of feedback loops can be avoided.

Approach 4: Reducing system criticality by applying an evolutionary algorithm

A fourth method of structural optimization, the evolutionary algorithm (EA) approach, was applied and is presented in Section 3.6. As the most efficient measures for reducing feedback loops were already on hand, the evolutionary approach had to deliver the same result by trying to reduce the overall system criticality. The evolutionary algorithm had to find and remove the two critical edges that are integrated in the majority of feedback loops (A-K, K-U). Secondly, the reduction of the feedback loops had to lead to a clearly recognizable reduction of the fitness index (which means a practical amelioration) in order to prove the applicability of this optimization criterion. The settings for the algorithm application were as follows: The ratio of attainability to closeness and of activity to passivity was set to 1 (see Equations 3-15 to 3-19); that means that these criteria were equally weighted. The only structural variation that could be performed by the evolutionary algorithm was the removal of dependencies. The results of the application are that the critical edges were reliably identified and removed in 25 runs with 1000 iterations each. That means that the result derived was quickly found and computational efforts were acceptable. Although no run-time optimization had been considered during the implementation, executing 1000 iterations required less than one minute on a standard PC. The overall fitness index was improved by 28% due to the removal of the dependency from “Element A” to “Element K”; additionally removing the dependency from “Element K” to “Element U” led to an improvement of the fitness index of 55%. If, however, the two dependencies, which have been the focus of the tearing approach, are eliminated instead, the fitness index can only be decreased by 28% (31% if dependency 3 from Figure 4-2 is eliminated as well). It must be noted that these values do not represent the reduction of the fitness itself, but of the fitness indexes. These do not represent absolute values, but permit the comparison of the initial structure and generated alternatives (i.e., a positive or negative evolution). The values are not applicable for absolute statements or the comparison of different structures. The results show that the structure evaluation and optimization by EAs can be applied, as the created fitness indexes rate the structures correctly. The analysis matrices of the EA application can be seen in the Appendix.

Conclusions

The DSM analysis of the packing machine illustrates that it is important to initially identify the structural criterion that represents the cause of the problem. The application of the partitioning procedure allowed three clustered areas of product components to be identified; however, this did not help solve the question on hand. In general, complex structures allow for a multitude of analyses, but it is important to select the relevant ones depending on the specific use case. The computer-supported, feedback-loop analysis resulted in a large quantity of circular change propagations. These could be identified as the structural representative of large change efforts and unsatisfying product robustness for component adaptations in practice. Consequently, the objective of structure optimization was the effective and efficient reduction of feedback loops, i.e., largest number of reduced feedback loops, by applying as

few product changes as possible. The propositions of optimization measures are determined by four methods that show different aptitudes for creating beneficial results. The tearing method is based on a preceding partitioning of a matrix depiction and can identify dependencies, whose elimination lead to a reduction of feedback loops. However, dependencies chosen directly depend on the quality of the matrix partitioning, which permits a multitude of different alignments in complex structures. Therefore, the appropriate selection of dependencies for the successive tearing approach (eliminating dependencies from the matrix) happens mainly by chance. Tearing aims primarily at the optimization of a modular design; the reduction of feedback loops is only a side-effect, because dependencies that reach back from downstream clusters are eliminated. A more appropriate determination of dependencies for elimination is to compute the dependencies which are most involved in the totality of feedback loops. Therefore, all feedback loops have to be identified and run through. This deterministic approach reliably points out the most appropriate dependencies for further optimization measures, but can result in high (or even non-executable) computational efforts if large quantities of feedback loops exist. The third approach to identify promising measures for feedback loop reduction is by the logical analysis of structural constellations. This does not require comprehensive computation, but powerful network visualization has to be available. This approach depends highly on the combined appearance of structural constellations and can not be transferred directly to arbitrary systems. Additionally, this approach requires intense knowledge and experience in structural analysis and can not be described in a generic procedure. Finally, a genetic algorithm was applied that uses a rating of the overall structure criticality for optimizing the structure. It can be proved that suitable results can be obtained that comply with the results found by other procedures; however, the applicability to further scenarios still needs to be investigated and depends highly on the settings chosen for the EA.

4.1.2 Structure analysis of an automotive engine

A further example of a structure describes the change dependencies between 28 components of an automotive engine. Figure 4-4 depicts the corresponding matrix alignment. In this case, objectives differ from those presented in the example of Section 4.1.1. The structure is not meant to be optimized by structure-based measures; rather developers want to gain a better understanding of the internal dependencies. That means that the objective of the structure analysis is to obtain a structure manual, which details structural characteristics and can be used for planning future product design.

Although the quantity of elements is almost the same as in the preceding example, a complete feedback loop analysis could not be successfully executed by available computational possibilities. Even if only the circular dependency paths comprising a maximum of 14 elements are considered, almost 300 000 feedback loops can be determined. Of course, the overall number of feedback loops could be computed, provided that required hardware equipment is available. However, the enormous amount will not allow a detailed consideration of selected circles. This outcome was rather surprising, and further analysis had to explain the existence of so many feedback loops. The DSM was realigned by applying a triangularization algorithm implemented by use of existing step-by-step procedures. The

resulting matrix can be seen in Figure 4-5. It shows three clustered areas indicated by the numbers 1 to 3. Within each cluster the amount of feedback loops is typically high (due to the high degree of connectivity). Although the clusters comprise up to ten elements each, the clusters' internal feedback loops can not be the only reason for the extremely high quantity of circular dependencies in the overall structure.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1 Comb. chamber		X	X	X						X	X				X	X		X			X	X					
2 Cylinder wall	X			X																								
3 Cylinder head	X								X	X	X			X	X	X					X	X					
4 Piston	X	X			X																							
5 Piston rod				X		X																						
6 Crankshaft					X		X	X															X					
7 Gearbox																												
8 Chain drive									X					X														
9 Var. intake cs			X					X		X									X			X	X					
10 Intake valve	X		X	X					X		X												X					
11 Intake port	X		X							X		X	X												X	X		X
12 Airjet											X												X	X				
13 Aspiration											X													X	X	X		X
14 Var. outlet cs			X					X							X								X					
15 Outlet valve	X		X	X										X		X												
16 Outlet port	X		X												X		X											
17 Manifold																X		X							X			
18 Exh. gas treatm.	X																X						X					
19 Pump									X											X								
20 Common rail																			X		X		X					
21 Fuel injector	X		X																	X		X						
22 Glow plug	X		X																		X		X					
23 Electric module									X			X		X				X		X	X	X		X	X	X	X	X
24 Throttle												X	X										X					
25 Exh. gas return											X		X				X						X					
26 Port shut-down											X		X										X					
27 Valve shut-down											X		X															
28 Control module											X		X										X					

Figure 4-4 Change impact within components of an automotive engine

It can be seen in the DSM of Figure 4-5 that cluster 2 overlaps with both of the other clusters; i.e., the dependency chains extend over more than one cluster. Whereas the clusters overlap in shared elements, dependencies can be identified that link between the clusters. These bridge edges are encircled in Figure 4-5 and denoted by the numbers 4 to 6. Above the matrix diagonals the index "a" is applied, whereas "b" indicates the corresponding bridge edges below the diagonal. Whereas dependencies in the circle of 5a link from cluster 1 to cluster 2,

dependencies in the circle of 5b link in the opposite direction. An interesting constellation is depicted by the two dependencies denoted by 4a and 4b. These represent the only direct link between cluster 1 and cluster 3. Even if the objective of the structure analysis is not the determination of optimization propositions, these two edges would possess high potential for structural improvements, as their avoidance would cancel the direct coupling of two clusters. The existence of the link in both directions further contributes to the huge amount of feedback loops. If only one of both dependencies were absent, feedback loops could not span both clusters (dependency chains propagating from one cluster to the other could not reach back again).

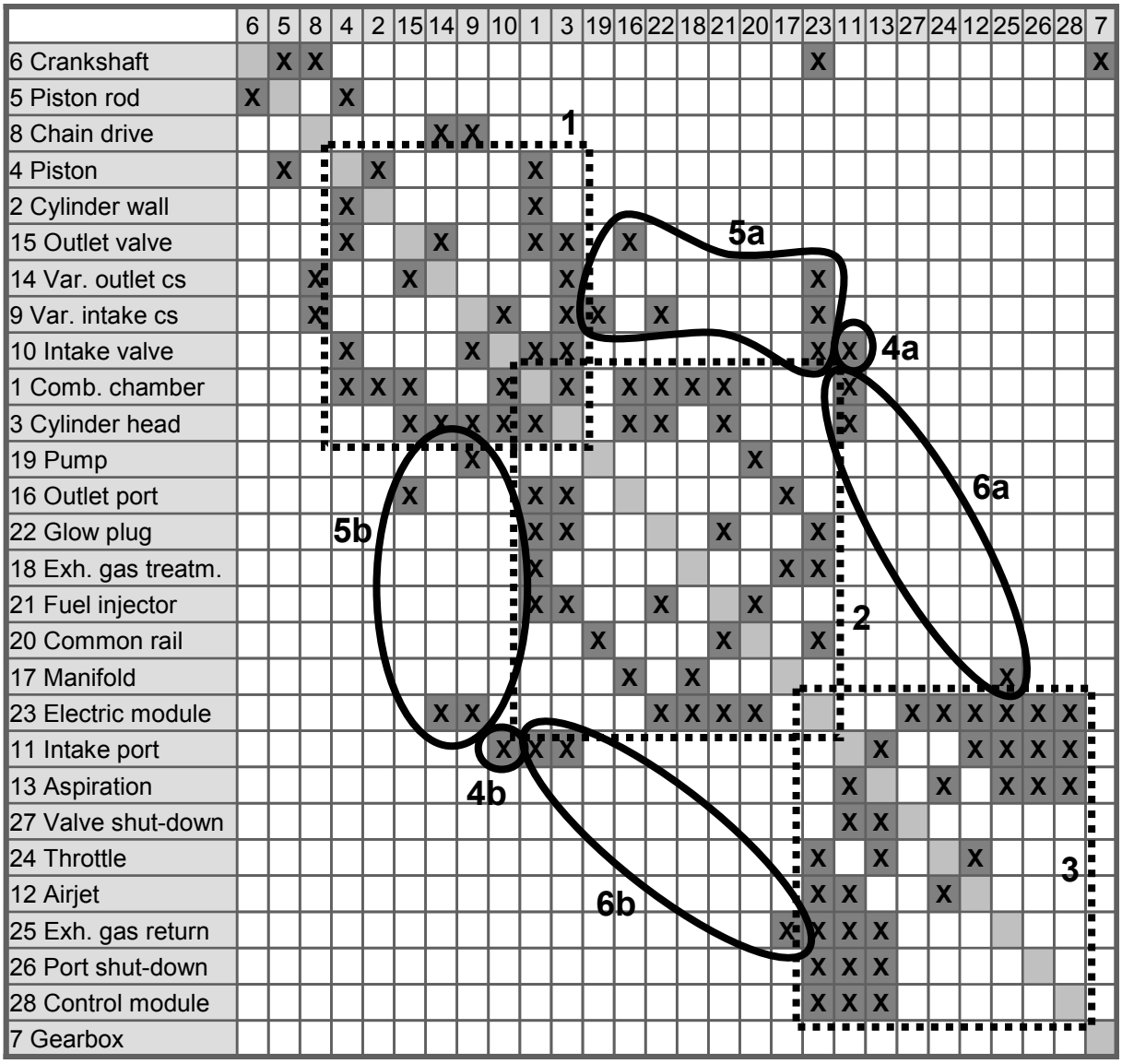


Figure 4-5 Clustered realignment of change impact within components of an automotive engine

Additionally, limits of the matrix visualization can be seen in the alignment of Figure 4-5. As is explained in Section 3.4.2 (see also Figure 3-25), elements that belong to more than two

clusters can not be positioned correctly. In the present use case the “Electric module” is depicted as a shared element of clusters 2 and 3. However, the “Electric module” further possesses three dependencies to elements from cluster 1. If the element were aligned differently, the assignment to cluster 3 would be lost. In any case, the “Electric module” represents a core element of the structure that constitutes part of all three clusters.

The application of further basic analysis criteria did not show significant results, because the overall connectivity in the structure is rather high. Generally, the more intensely elements are linked, the more difficult is it to identify meaningful structural constellations. For example, 27 hierarchical subsets could be identified, but they span most of the structure. In contrast, the consideration of the structural embedding of selected elements possesses a high expressiveness in the present use case. In interviews the developers remarked about the major importance of the “Electric module” and the “Combustion chamber”. Therefore, their structural surroundings, i.e., the directly connected elements, were isolated for closer consideration. Figure 4-6 depicts both subsets for a comparative analysis.

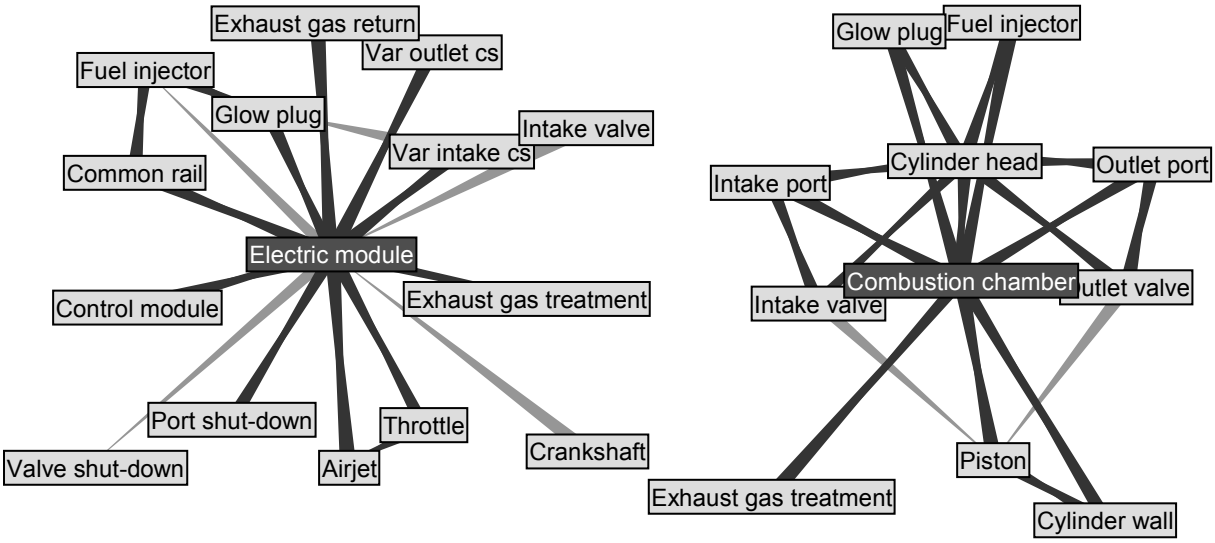


Figure 4-6 Significance of the structural embedding of selected elements

At first glance, both subsets look different. Whereas the area surrounding the “Electric module” appears mostly star-shaped, the area surrounding the “Combustion chamber” is highly cross-linked. The behavior of both elements is different in case of change impacts to the system. If, for example, the “Port shut-down” is adapted, in the first instance, only the “Electric module” will be affected, as indicated by the connected dependency. If further change propagation can be stopped by appropriate technical measures, no further elements of the system will have to be adapted. In practice, for example, a software update of the “Electric module”, could permit the integration of a changed “Port shut-down” without further need for updates to other system components. Different consequences arise from a change impact to a component in the area directly surrounding the “Combustion chamber”. If, for example, the “Outlet port” were adapted, a direct impact not only affects the “Combustion chamber”, but also the “Cylinder head” and the “Outlet valve”. These components, however, also affect the

combustion chamber, i.e., the initial adaptation of the "Outlet port" produces three different impacts on the "Combustion chamber" (even more impact chains exist, if not just indirect connections passing one element are considered). If these three adaptation requests are mutually contradicted in practical application, the system adaptation can become difficult. In conclusion, when considering these two subsets, adapting elements in a highly cross-linked change dependency network can not be recommended. In many cases, one required functional adaptation can be realized by different adaptation measures. A measure should be chosen which adapts an element that is integrated in a linear or star-shaped structure.

4.2 Multiple-domain approach

Whereas the two preceding examples focused solely on the analysis and optimization of a DSM structure, in the following a comprehensive MDM approach will be presented. The case study considers the development of a race car by the student group TUfast[37]. For the competition, the car design evaluated handling, performance, engineering and design to cost from different perspective. Detailed information on the project can be seen in Section 1.2.

The management of complex dependencies is important in the development of each new car design on the product level as well as in the development process and the organizational structure. The race car's concept has to be fine-tuned to be equally successful in all tests prescribed by the regulations. Numerous compromises have to be made, implications of a single design decision extend to other parts, and a robust system forms the basis for meeting all targets. The underlying design process and organizational structure are, in many aspects, similar to the automotive industry. The design process is highly distributed, involving about 25 engineering students in a concurrent design environment. Typical challenges are apparent, such as the management of releases, the modularization of the product as well as the development process, and an optimized distribution of information. The TUfast group relies on rapid development processes (i.e., time constraints) because a new race car has to be developed once a year. Although experience from previous designs is useful for next year's development, repeated fluctuation of team members raises difficulties for the group. Against this background, structural information was collected in order to identify critical constellations and provide suggestions for optimization.

System definition and information acquisition

Five domains were identified during the system definition of the structural scenario: components, people, data, process steps, and milestones. Some sub-matrices could be directly recorded, either because information was already available from other modeling approaches or because people could easily denote the dependencies. These available matrices are shaded grey in Figure 4-7, and the associated dependency meaning for every subset is given (the matrix has to be read as "row influences column"). Altogether 14 different matrices were available as native data, eleven represent inter-domain (DMM) and three represent intra-domain (DSM) matrices. The two matrices shaded in light grey have not been acquired

[37] TUfast: http://www.tufast.de

directly but represent inverted matrices of information already available. For example, the DMM that links from “People” to “Components” (“People work on Components”) can be derived directly from the linking between “Components” and “People” (“Components are processed by People”). That means that the matrices shaded in light grey are on hand, but do not comprise supplementing information to the opposite matrix.

	Components	People	Data	Processes	Milestones
Components	Change	Processed by	Repre-sented by		Completed at
People	*Work on*		Generate	*Work on*	
Data	Required for	Required by		Required for	Available at
Processes		Executed by	Generate	Transfer information to	Completed at
Milestones					Followed by

Figure 4-7 Determination of involved domains and dependency types

The initial task is to permit a better general understanding of the complex development process. For this reason, the opportunistic analysis approach was applied (see Section 3.2 and Figure 3-20). Based on the available 14 sub-matrices, DSM networks for all five domains could be derived by combinatory MDM computation. Some of these derived DSMs were subsequently analyzed in detail. Black arrows in Figure 4-8 show the computational logic of the selected matrix computations. Component and process DSMs already existed as original data (round arrows) and were therefore considered for direct DSM analysis. Five other networks were each computed from two related MDM subsets according to the straight arrows. These are assumed to provide the most significant results. Some available matrices were not used at all due to the low quality of the input data. That means that the information base of interrogated people was generally vague for the established dependencies (e.g., for the matrix of linkages from data on components).

In theory, the available information would allow for 41 different DSMs to be derived. Obviously, this is not recommended, as the sheer amount of derived DSM networks can scarcely be handled for subsequent analysis and interpretation. By considering the significance of resulting networks, the following DSMs were prepared for further analysis:

- Native DSM of components linked by change impact;
- DSM of people dependencies based on the DSM connecting within the components and the DSM linking components on people; people become linked, if they are responsible for components, which comprise a mutual change impact;
- DSM of people dependencies based on DMMs between data and people; people become linked, if one person provides data that is required by another person;
- DSM of data dependencies based on DMMs between processes and data; two data sets become linked, if one is required for a process that generated the second data set;
- Native DSM of process steps linked by information transfer;
- DSM of process steps dependencies based on DMMs between data and processes; process steps become linked, if one generates data that is required for the second process step;
- DSM of milestones' dependencies based on the DSM connecting within the process steps and the DMM linking process steps on milestones; two milestones become linked, if they have assigned process steps that comprise a mutual information transfer. Normally, milestones are defined sequentially; if their alignment is determined by use of the related process steps, the milestone dependencies can be used for verification of a correct time line in process planning.

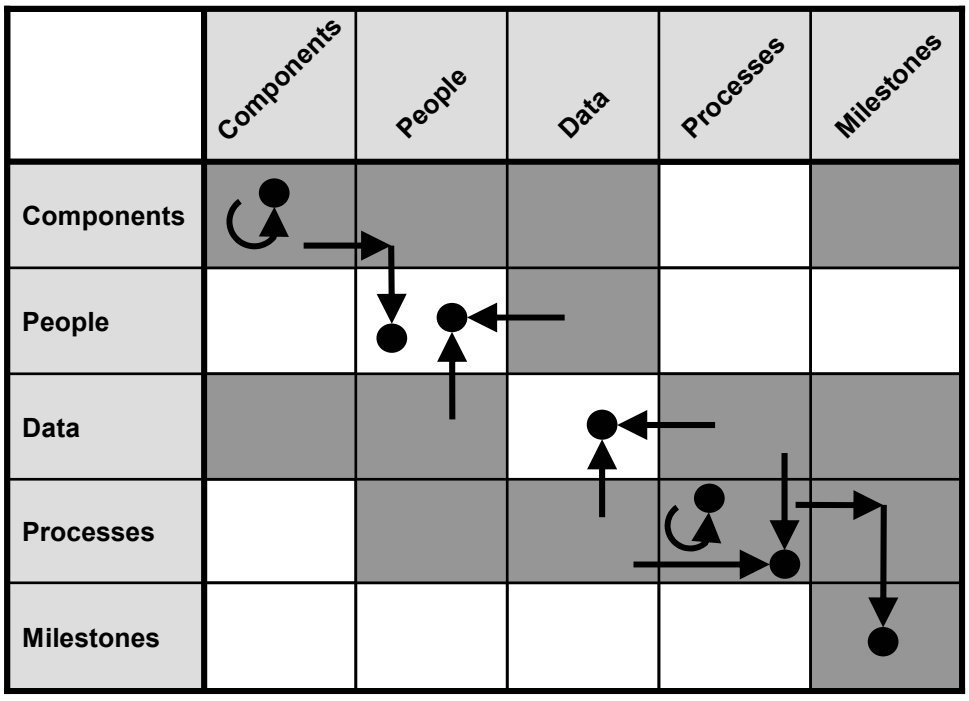


Figure 4-8 Determination of DSMs from available MDM subsets

Structure analysis and interpretation

To consider this use case, a general system level is applied. System domains contain between 10 and 30 elements each, corresponding to reasonable efforts for information acquisition and analysis. In the following, the analysis and interpretation of three of the prepared DSM networks will be shown in detail.

Native DSM of components

First, the analysis of the component-based DSM representing a change impact within components is executed. As this component network also serves as an information base for a people DSM derived later, it is useful to be familiar with this structure first. In general, component networks are of major importance in considering complex product development. Each product is built up by components that incorporate functions and requirements and call for adequate processes, e.g., in their development or production. Figure 4-9 displays a network overview in a matrix depiction, whereas Figure 4-10 depicts the associated strength-based graph.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1 Motor contr. unit		X			X				X																		
2 Diff. control unit																											
3 Data logging																											
4 Wiring harness																											
5 Radio module																											
6 Engine	X						X	X	X	X	X					X											
7 Airbox	X																										
8 Exhaust system	X															X											
9 Gasoline supply																X											
10 Oil supply																X											
11 Cooling system																X											
12 Gear unit													X		X												
13 Differential																											
14 Shaft drives													X														
15 Chain drive												X	X	X													
16 Frame/monoc.						X	X	X	X	X	X		X	X			X		X	X			X		X		X
17 Crashbox																X											
18 Gear shift																											
19 Pedals						X										X											
20 Steering wheel																					X		X				X
21 Wheels/tires																X											
22 Wheel trunk																					X					X	
23 Guide																X						X		X	X		
24 Suspension																							X				
25 Stabilizer																X											
26 Brakes																					X	X	X				
27 Steering																X				X							

Figure 4-9 Matrix visualization of the component structure

In the network, two strongly connected parts can be found that contain 18 and 2 nodes (element indexes: 15-12; 19-17-11-10-8-7-9-1-6-25-24-22-23-20-27-16-21-26). Three elements ("Data logging" (3), "Wiring harness" (4), and "Gearshift" (18)) are completely isolated from the rest of the structure (not shown in Figure 4-10), and therefore do not provide or receive any change impact. "Differential" (13), "Differential control unit" (2), and "Radio module" (5) depict end nodes, which only receive change impact from other components but possess no active impact. The "Frame/monocoque" (16) forms the central active as well as passive bus element that connects to the majority of further components. Additionally, the "Engine" (6) can be seen as an active bus element, because its adaptation can affect seven other components. Considerations as to the closeness and attainability criteria result in the conclusion that the "Brakes" (26) receive the most indirect change impact and the "Airbox" (7) spreads the most indirect change impact. Altogether, the structure possesses only a small degree of connectivity that facilitates computational analyses.

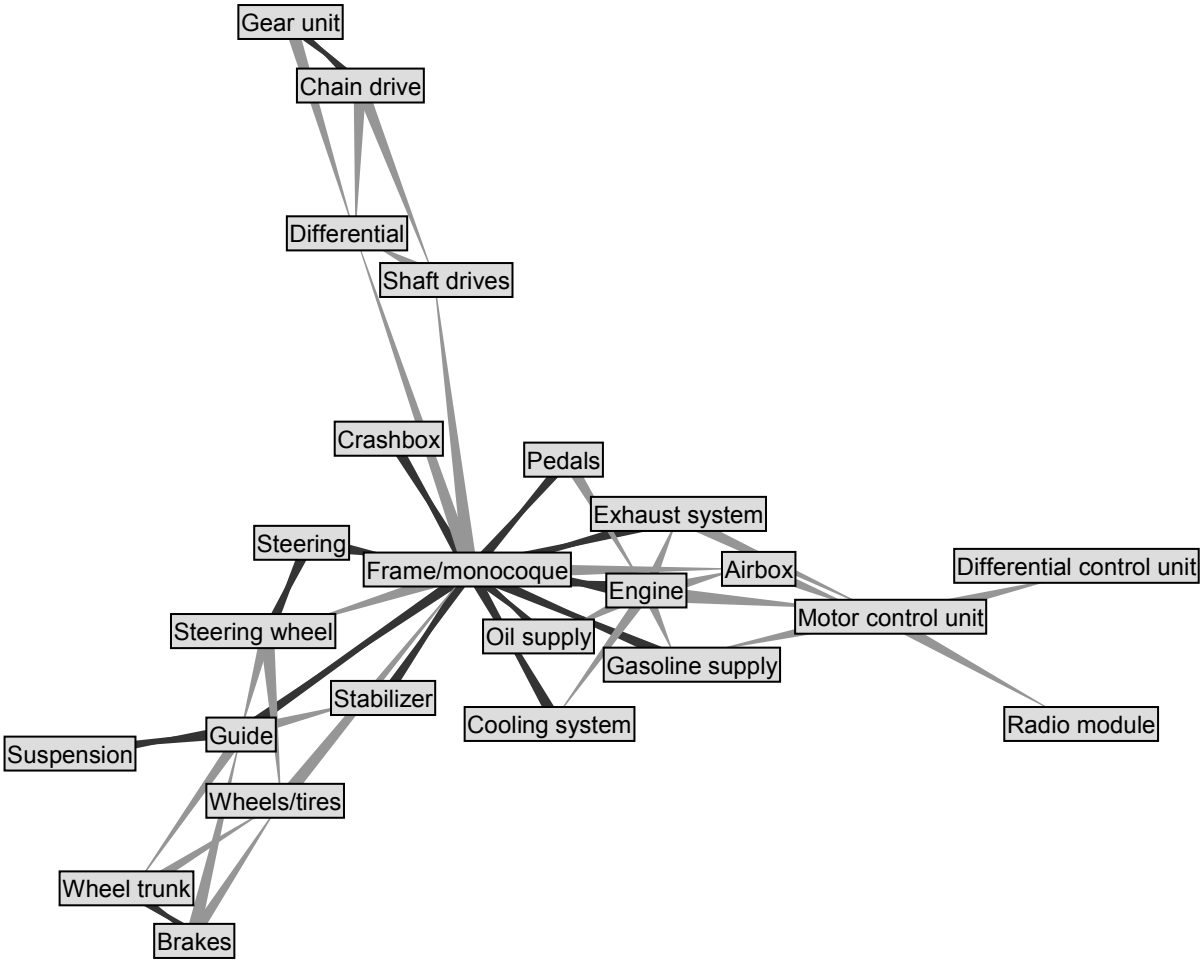


Figure 4-10 Graph visualization of the component structure

The depiction in Figure 4-10 by a strength-based graph allows further structural characterizations. The advantage of this visualization is the automatic alignment of components, which is intuitive for users. Graph nodes repel each other and are mutually

attracted by graph edges. It is possible to assign different weights to graph edges, which result in different attraction forces (leading to different element distances) between nodes. From the alignment in Figure 4-10 it can easily be seen that the entire component structure is composed of three major blocks coupled in the central component “Frame/monocoque” (16). The “Guide” (23) and the “Motor control unit” (1) constitute articulation nodes, i.e., they represent the only connection between two sub-graphs. Bi-directional dependencies are pointed out by dark grey arrows in Figure 4-10. Most of these mutual impacts between components concern connections with the “Frame/monocoque” (16).

Next the matrix depiction was realigned by applying a triangularization algorithm in order to identify modularized and hierarchical subsets in the structure. Figure 4-11 shows the resulting alignment and indicates the most interesting subsets. Four clusters could be arranged, and implied elements can be classified by a comprehensive meaning. The elements belonging to the “Frame & chassis” form a cluster of seven elements, which are for the most part equally connected and possess several bi-directional dependencies. Elements belonging to the input side of the power train are grouped together and represent a small hierarchical subset with the “Engine” (6) as the top element (in this hierarchy, the “Pedals” (19) are located on top of the “Engine” (6); however they only posses one dependency that links to the “Engine” (6)). The elements belonging to the output side of the power train form a cluster, whereas the two mutually linked elements “Chain drive” (15) and “Gear unit” (12) can be taken as their own block which describes the energy conversion. The “Frame/monocoque” (16) is linked to all identified blocks in the structure except for the “Conversion” block. The “Guide” (23) is connected to the “Frame & chassis” block and the “Power train (input)” block. Therefore, the “Frame/monocoque”, (16) as well as the “Guide” (23), represents generally connecting components in the entire structure.

Seven hierarchical subsets exist in the structure, which can not be indicated simultaneously by the matrix alignment. Algorithmic analyses show that these hierarchies comprise 129 edges and 125 nodes in total. Hierarchies in component-based change impact networks mean spreading the effort of adaptations. For this reason, structural optimization should aim at the minimization of these subsets. Further search for basic structure criteria did not result in more significant conclusions, but supported the characteristics already found. For example, a feedback loop analysis identified 46 cyclic change propagations; in these subsets the element “Frame/monocoque” (16) was included 41 times.

The findings provide interpretations and suggestions for subsequent component development. The development of all elements belonging to the same block must be coordinated when elaborating design details. As the blocks represent the technical achievement of comprehensive product functionalities, they already correspond to the general project organization. The block-internal degree of connectivity is rather high, thus a coordinated development is required and should be undertaken by the person in charge for the functional realization.

The person responsible for the development of the “Frame/monocoque” (16) must be included in each bi-directional coordination task. Due to the dominant cross-linking of the “Frame/monocoque” (16), it is highly probable that adaptations to any other components will

affect it. Therefore, the direct integration of the related person in charge can help to shorten iteration loops and to minimize rework.

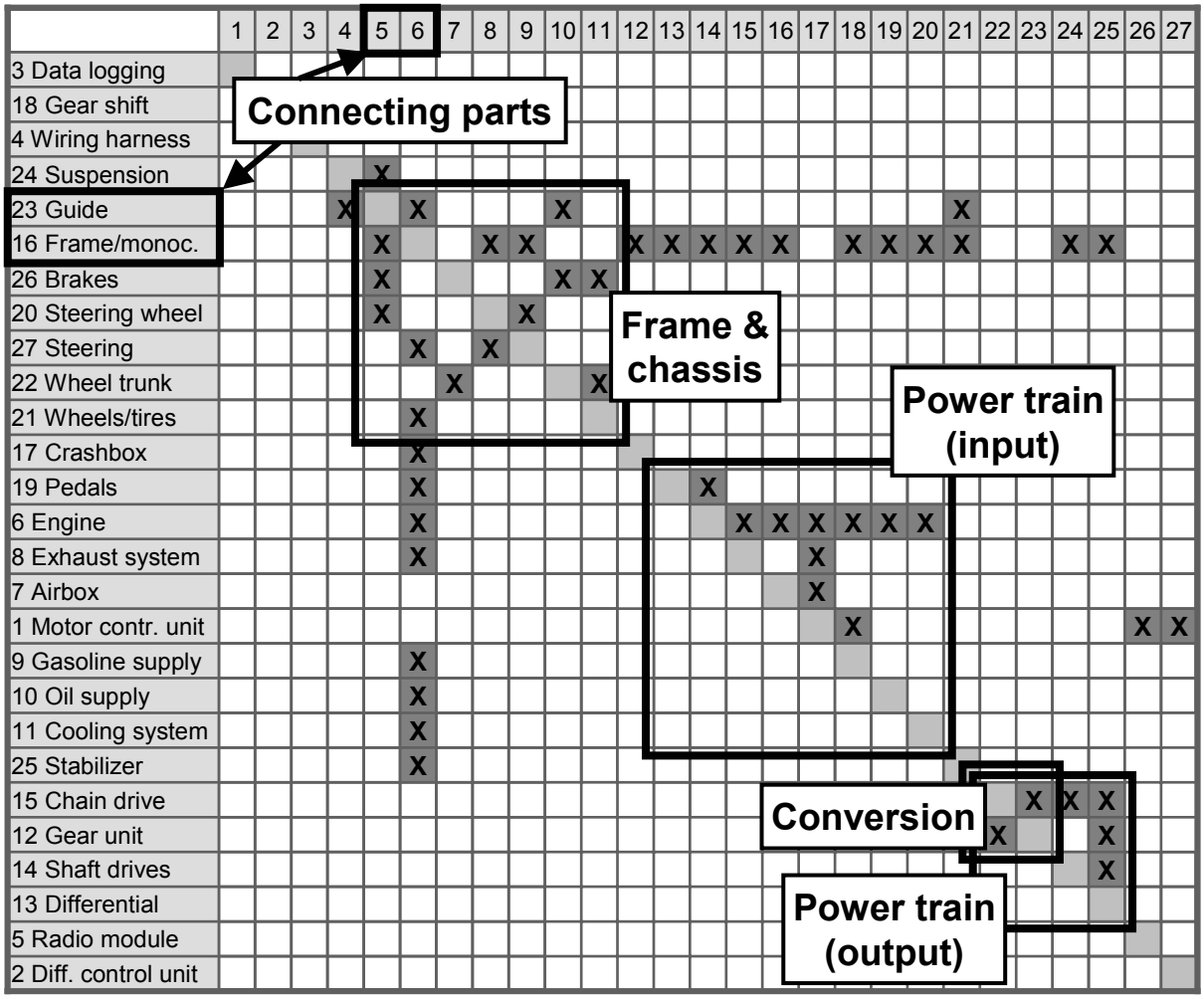


Figure 4-11 Triangularized component DSM with assigned significance of subsets

Changes in one of the strongly connected parts can be conducted without having to consider the other one to any great degree. Inside both strongly connected parts, by definition all elements can be reached by at least one dependency path from all other ones. Two elements belonging to different strongly connected parts can still be connected by a dependency path, but a cyclic dependency can not exist between them. Consequently, element changes can not result in self-energizing or self-impeding effects that would be difficult to manage. Additionally, all components represented by isolated nodes can be adapted independently from all other components of the structure. In project management the development of these components can be planned to be executed simultaneously with any other development task.

Generally, the avoidance of network dependencies (or even network elements) can help reduce the existing hierarchical subsets. In the case of change impact between components, this means, on the one hand, that development resources are required. On the other hand, standardization of the present component interfaces would also reduce dependencies without

producing the need for any development work. Unfortunately, this decreases the flexibility in construction. By standardizing the component interfaces almost every dependency can be avoided, but developers must be aware of the fact that simplification of the development process by minimized change propagation is attended by restrictions of component design and quite possibly suboptimal development results as well. For this reason, the reduction of undesired hierarchical subsets must be realized by the most efficient (and the least quantity of) dependency eliminations. In case of the component network presented here, the avoidance of three dependencies ("Frame/monocoque" (16) to "Guide" (23), "Steering wheel" (20) to "Guide" (23), "Guide" (23) to "Frame/monocoque" (16)) reduces the entire quantity of hierarchical subsets from seven to four, and approximately 50% of the components and dependencies primarily involved are no longer included. Most other eliminations of dependencies would only result in a reduction of approximately 10%, but would require extensive development resources and would restrict the flexibility of product design. In the following, all three dependencies identified must be examined to determine their possible avoidance. If these dependencies can not be eliminated by appropriate design change, further dependencies must be identified that are next in the rating of effective reduction of hierarchical subsets. This procedure equals a structure-based Pareto Analysis[38] and is described as structural ABC-analysis in Section 3.6.

Further recommendations can be made in order to improve the organization of the component-based design process: The need for coordinating component adaptations can be reduced if people working on several components are assigned to ones located in the same structural block. Team organization becomes easier as the block's internal need for coordination is greater than its external one. Consequently, small blocks consisting of two components, for example, should be assigned to only one person. As a matter of fact, component adaptation and resulting change impact cannot be avoided in the project considered (e.g., because of annual changes in the competition rules). For this reason, some components have an enormous impact on others (e.g., "Frame/monocoque" (16) and "Engine" (6)) require fast and efficient communication of conducted adaptations). Change processes concerning these elements should be supported, for example, by checklists or visualization of affected components (subsets extracted from the graph representation). It is possible to reduce the quantity of change impacts if the "Radio module" (5) and "Differential control unit" (2) (representing end notes in the structure) as well as the "Airbox" (7) (generally an indirect impact to other components) become defined late in the design process. In contrast, the "Brakes" (26) (representing a start node in the structure) should be specified early in the design process, because impact by other components is not anticipated.

DSM of people dependencies based on component dependencies

After executing the analysis of the native component structure of the use case, further dependency networks that are derived by the multiple-domain approach are analyzed. Generally, people often know about specific structural characteristics, even if the far-reaching

[38] The Pareto Analysis is used for the elementary identification and classification of relevant objects in a group ([DAENZER & HUBER 1999, P. 430FF], [BLANCHARD 2004, P. 451]).

or overlapping impact is not expressed explicitly. However, the dependencies between people are not available explicitly and therefore have to be derived from component and data views. First, a network connecting people is computed by the dependencies between people and components. The computational scheme corresponds to case 4 described in Section 3.2. The dependency meaning of the resulting network is the fact that two people are interrelated, because they work on (different) components that impose a change impact on each other. The direction of a dependency between two people corresponds to the direction of the change impact between the associated components. Different weights of dependencies between people may result if two people are linked to each other by more than one component change impact (resulting in different attraction forces when depicted in a strength-based graph). Figure 4-12 illustrates a part of the resulting intra-domain network in a strength-based graph; this representation focuses only on bi-directional dependencies between people. People located close to each other work on several components which are dependent on change impact (i.e., adaptation of the first component requires adaptation of the second one). Consequently, the closeness of people in the graph can be interpreted as the importance of cooperation, because the closer two people are located to each other, the more the mutual change impact affects them. Nine complete clusters exist in the entire component-based people network; each cluster comprises three people. “Bernhard” and “John” are involved in all of these clusters. In addition, a large quantity of feedback loops exists in the structure; “Dan”, “Bernhard”, and “John” are included in most of them. Furthermore, 15 hierarchies can be identified in the structure, comprising between 14 and 18 people mostly on three hierarchy levels.

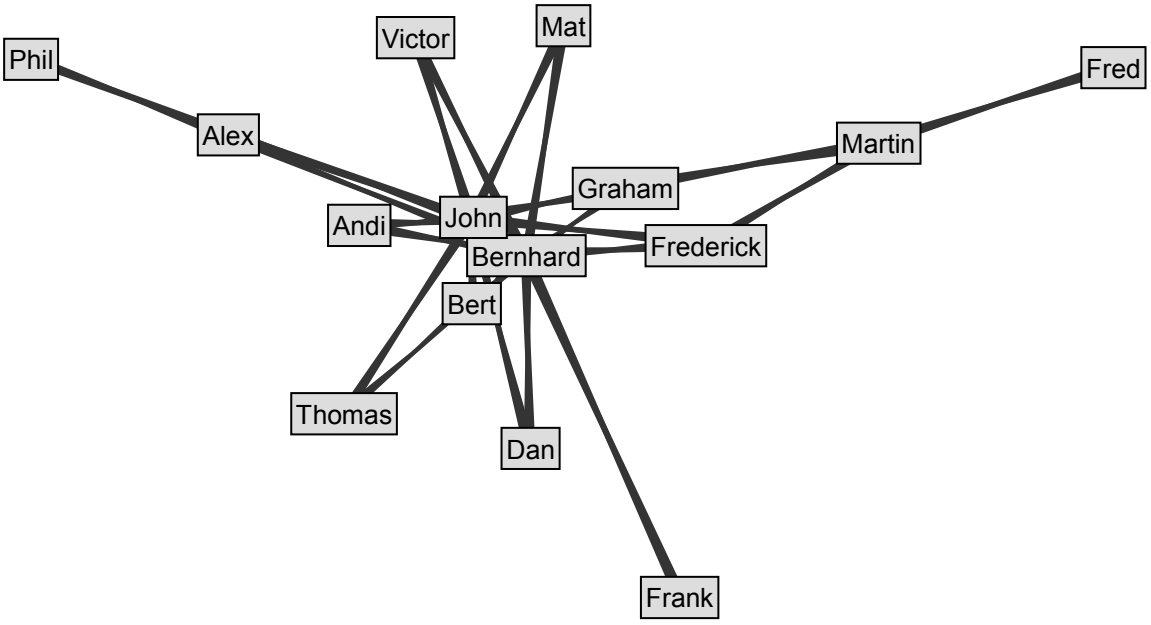


Figure 4-12 Sub graph of bi-directional dependencies between people based on people’s links to components

In the practical development process thus far, all team members take part in the weekly team meeting; however, Figure 4-12 suggests reconsidering people’s coordination. Many team members require bi-directional coordination that can not be perfectly executed in the general

meeting (or else it becomes too time-consuming for the other members). A lot of face-to-face coordination must be done in cooperation with the two persons most involved, “John” and “Bernhard”. If the required discussions are done one after another in the general team meeting, these are irrelevant (but time-consuming) for all other team members, who are not affected by the specific coordination tasks. Whereas a few people have a major need for internal as well as external coordination, others only require direct contact with one person. The decision to arrange a team meeting to include everyone probably arose from a lack of knowledge about the explicit coordination needs. Therefore, it would be useful to prioritize a bi-directional exchange based on the quantity of the component change impact. People should ensure that they possess adequate means of communication, when they are located at a network position that requires major coordination.

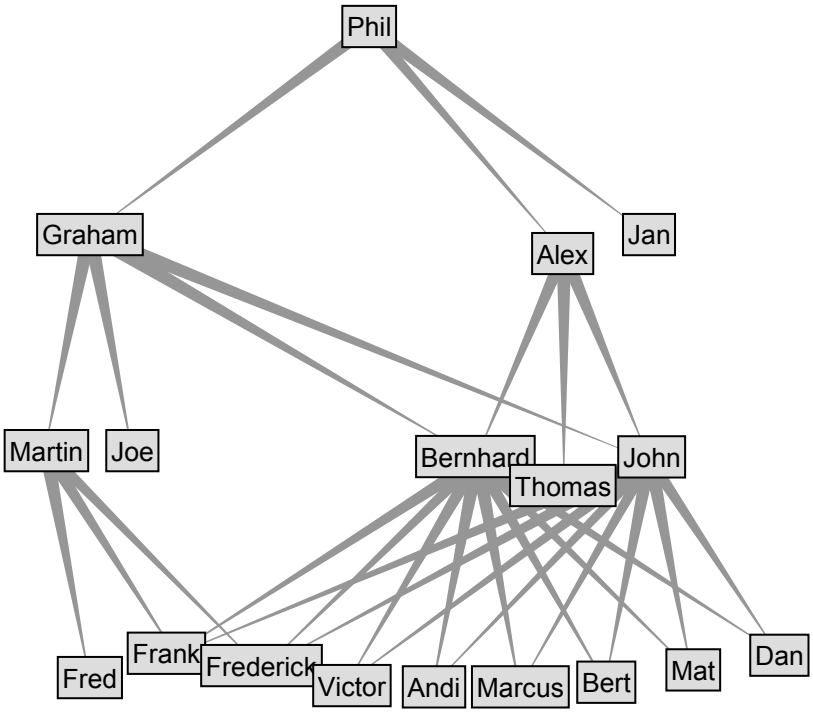


Figure 4-13 Hierarchical sub graph filtered from linking of people

Figure 4-13 shows the structural relevance of hierarchical subsets in the people network derived. The illustrated sub graph represents one of the 15 hierarchies included. Here, a hierarchy illustrates how communication needs to propagate due to component adaptation initiated by one person. In Figure 4-13, “Phil” represents the top element (initial point) of the hierarchy. If one were to consider only the people within his direct surroundings (nodes connected by dependencies), the situation might be underestimated, as only three people are directly affected due to the component change impact. One may assume that component adaptations executed by “Phil” would only have to be agreed upon in a meeting of four people (“Phil” together with the three other directly linked developers). However, change impact caused by “Phil” may indirectly affect up to 17 other developers, if subsequent dependencies are taken into account. Only if the three directly affected developers can reliably rate the

relevance of an adaptation for developers who may be indirectly affected, can decisions on "Phil's" adaptation be agreed upon in the meeting of four people. A useful method for practical application is to filter the paths of change propagation from the hierarchical subset which may cause the highest change impact. In the example in Figure 4-13, these would be the linking paths from "Phil" to "Bernhard" and to "John" (passing through dependencies to "Graham" and "Alex"). If changes executed by "Phil" were not propagated by "Bernhard" and "John", six other people would not be affected. If these paths are collected in a checklist, users can easily verify whether a currently planned component adaptation will cause wide-spread impact (and may consequently avoid it or organize the required resources for it). Such checklists have to be deduced individually for every person who represents the initial point of an identified hierarchy and must be integrated in the structure manual.

DSM of people dependencies based on data dependencies

A further important network represents the linking of people because of their data dependencies in the project. This can be computed by the two available DMMs, which connect people and data. Here, the dependency meaning is "person generates data" and "data required by person". The resulting dependency meaning in the DSM is as follows: "a dependency is directed from one person to another if the first one generates data the second one requires". The arising network is displayed by the strength-based graph in Figure 4-14. Although it seems to be somewhat complicated (due to the high quantity of dependencies), the structure is highly significant and allows important interpretations. Generally, two different groups of people can be identified (with only few outliers) within the representation. The first group is located in the middle of the graph, possessing intensive active and passive interactions concerning data. These people require large amounts of data in the design process and generate relevant data for the developers as well. The second group of people is characterized by their strictly passive dependencies in data exchange. These people require data as input specification for their development tasks. The work results are not required by other developers and therefore do not get fed back into data descriptions. There are only a few people who do not belong to one of these two groups.

Possible optimization measures concerning the development process are as follows: On the one hand, people who generate data should know who has to receive their output. This could be assured by visualizing the data recipients surrounding each individual. If this information is available from a structure manual, this facilitates the push principle of information flow. On the other hand, people depending on data provision should know about their information suppliers. Visual support of individual supply chains in a structure manual can permit a pull principle that becomes important in the case of supply delay, for example.

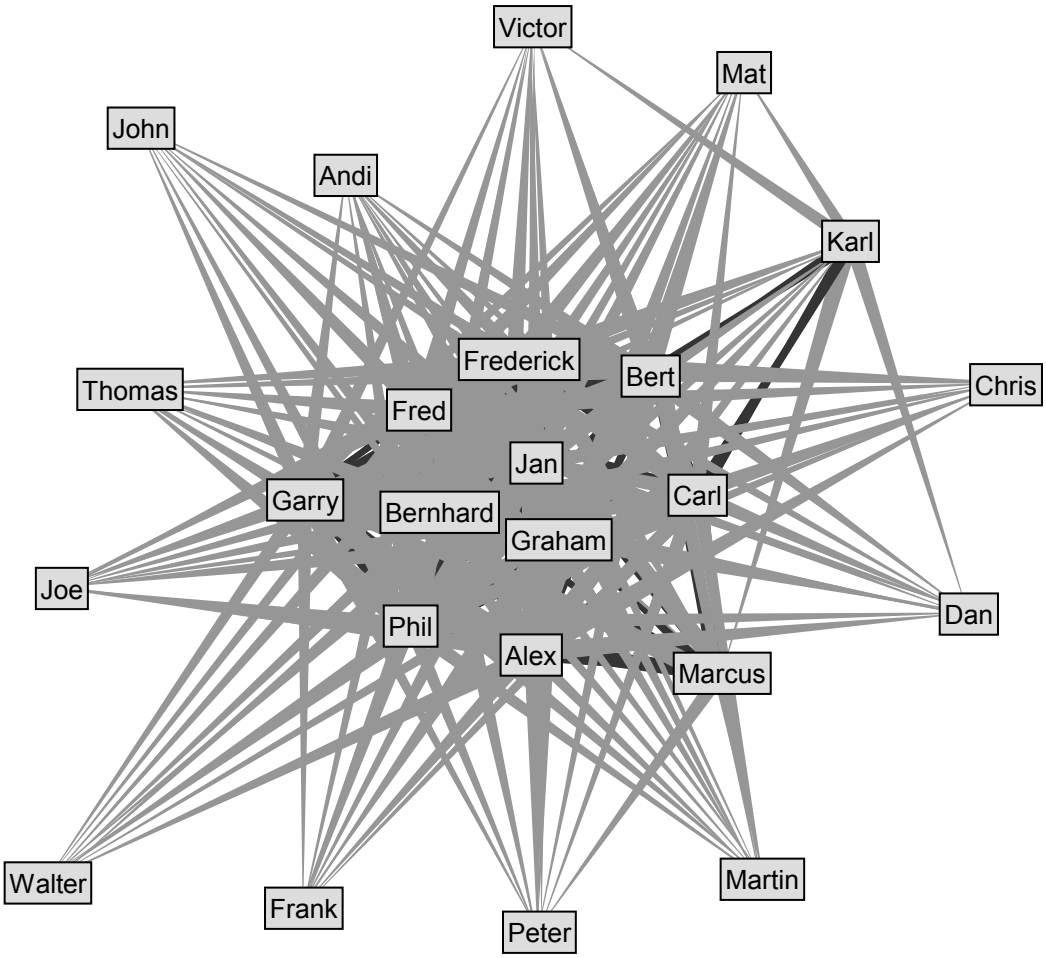


Figure 4-14 Dependencies between people based on their links to data

The network of people based on their data exchange allows for the fluctuation in the team members to be anticipated, which poses a major task in the student race car team. If a person belonging to the passive data receivers wants to quit the team the situation is comparatively easy. Of course, the team management has to find another person executing the related tasks, but the information flow will not be disturbed; other people do not depend on information from the team member in question. The situation becomes much more complicated if a team member from the inner group of the graph representation changes. In this case it is not only necessary to find another person for processing incoming data and for attending to the person's tasks, it must be guaranteed that the new team member knows about his obligation to provide data to other team members. If this new team member is not introduced correctly, major disturbances of the design process may result due to lack of data. For these reasons, it is recommended that a structured plan for the replacement of such team members be set up.

Summary of the case study

The case study showed the application of the multiple-domain approach for the complex structural dependencies within a comprehensive development project. The system definition

initially executed resulted in a MDM layout comprising five domains. Information acquisition was executed efficiently, as many dependencies between system elements were imported directly to the MDM. The resulting MDM subsets allowed 41 different DSMs to be derived; for subsequent analyses, significant networks were selected. Therefore, the dependency meanings that arise from logic DSM compilation were considered regarding the specific requirements of the use case. The DSM analyses subsequently executed resulted in an improved understanding of project dependencies, and several findings could be implemented for the development process.

Whereas most direct technical dependencies between product components were known by the developers, the structural embedding of components in the entire network of change impacts was for the most part unknown. This network provided new insight and allowed several organizational and process-oriented rearrangements to be developed. The further analysis of specific components led to their new classification in the development process.

Two of the derived DSMs represented dependencies between people and showed potential for improving the organization of team meetings and for minimizing problems due to the frequent change of team members. Formerly, only general team meetings with all team members were arranged, as the need for the mutual exchange between developers could not be explicitly specified. This deficiency was solved by the analysis of people's dependencies, based on their relation to components. Specific need for mutually coordinating people could be indicated and serve to establish more efficient individual meetings.

Additionally, people's dependencies due to their requests for and provision of data were derived from available MDM subsets. This allowed developers to be classified by data recipients and data providers. Depending on people's affiliation with these groups, different tasks could be defined for the improved organization of team changes that frequently occurred.

The complex structural dependencies of the case study reached across several domains and different types of dependencies. Therefore, it was not possible to directly apply analyses to the structure but to classify structural content by a MDM layout. Considerations as to the obtainable dependency meaning of derivable DSMs allowed networks to be selected, whose closer analyses provided significant details for improving actual problems in the development process. The case study showed that this meaningful derivation of selected subsets from the MDM represents a major challenge, as it is decisive for the findings subsequently obtained in DSM analyses.

5. Conclusions and outlook

5.1 Conclusions

Product development is characterized by a steady increase in complexity. Impact can be identified in all different areas of product development, such as markets, products, processes, or organizations. The different areas are highly interconnected, and therefore call for a holistic approach. Considered from a system perspective, the structures emerging from elements in all different areas and the linkages between those elements provide insight into systems' characteristics and behavior on an accessible and generic level. System structures are therefore the main object of consideration when attempting to comprehensively manage complexity in product development.

Reasons for the increase of complexity in product development are numerous. Market complexity and newly arising or changing market requirements pose a main source for required adaptations in product design. Due to the interconnectivity of the named areas, these requirements impact not only products but processes and organizations as well. Successful market offers have to meet these requirements and thus demand a variety of complex products and adaptations to them, which have to be implemented at low cost and within a short time to market. This trend towards increasing product customization is met by promising business strategies like Mass Customization or the concentration on niche markets, according to the long-tail phenomenon. In order to permit an application of these strategies in the context of technically demanding products, improved possibilities for a comprehensive approach to managing complex system structures are required but have not been available so far.

Existing methods aim at single aspects of the areas or domains mentioned and provide limited possibilities for comprehensive analysis, control and optimization. Due to the importance of domain-spanning dependencies, i.e., relations between product components and process steps for example, few methods have been developed that allow their mutual linking. Such methods are normally restricted to considering a combination of only two different kinds of fragments such as components and process steps.

The present thesis proposes an approach to meet both the major shortcomings of existing methods – the lack of promising and generic instruments for analysis, control and optimization of complex system structures on the one hand, and the applicability of comprehensive and domain-spanning problems on the other. The approach is founded on established matrix-based methods, and therefore provides the derived Multiple-Domain Matrix as the core of a systematic methodology. The approach supports the initial definition of relevant system aspects and the systematic acquisition of information on dependencies in the first steps. The possibilities of network visualization in graphs and matrices for reasons of information acquisition and system interaction have been enhanced in order to allow for better system understanding and user interaction. Available analysis methods and strategies for complex system structures have been transferred from the integrative fundamentals of graph theory and systematically complemented. Building on that, practical procedures for the analysis, control, and optimization of complex system structures have been generated,

culminating in the definition of comprehensive strategies for the management and design of complex systems.

The approach presented has been developed with iterative and recursive application to different areas of product development in industrial research projects. The interpretation of the different enhanced criteria for system analysis, in particular, has been elaborated by use of case studies. The complete approach has been validated by application to a comprehensive development project and allowed the derivation of different insights and improvements in different areas. First, better overall system understanding for the developers was achieved, and the findings from structure considerations resulted in an improved product design that is robust for required system adaptations. Secondly, the analyses of domain-spanning dependencies in the project enabled developers to improve project coordination and communication between team members and shorten development time.

Recapitulating, the approach presented demonstrated its suitability for the management of complex system structures in the field of product development regarding its practical applicability as well as obtainable results. It offers a sound basis for future enhancements concerning the management of further aspects of complexity as well as the methodology presented.

5.2 Outlook

As it is described in the present work, the effective application of the new approach for complex networks asks for intense software support. During the development of the methodical basis, some aspects were covered by prototypic implementation, but an integrated software solution according to the described requirements has not been available so far.

Effective data representation, which demonstrates the interface for users to interact with system structures, is decisive for obtaining a better system understanding and practical applicability of the approach. In the present work, a combination of matrices and strength-based graphs was chosen for the representation that seems to be best suited for the tasks considered. However, the limits of the chosen visualization techniques have also been pointed out. In particular, the possible misinterpretation of element alignment in matrices and graphs can result in applying wrong measures of management or structural optimization. Therefore, future development should also focus on the enhancement of visualization techniques, which improve intuitive understanding of complex structural dependencies and additionally minimize the limitations in the depiction of structural content. Starting points could be, for example, the implementation of a three-dimensional graph depiction or dynamic structure models, as further degrees of freedom could clarify representation. In general, a combination of established techniques of visualization appears to be promising.

Further possibilities to enhance the approach presented exist in the methodology of the Multiple-Domain Matrix concerning data processing. The computation of DSMs presented only considers directed dependencies between system elements. An extension to non-directed or even both-sided dependencies occurs in practical applications (e.g., the *actio est reactio* principle when considering mechanical forces). This enhancement would notably increase the number of deducible computational schemes. In order to assure usability, the dependency

meaning of all these schemes must be clarified; this is required to make use of interpretations. It is conceivable that DSM computations are not executed only by use of one supplementary domain (e.g., component dependencies by use of functional linkages), as discussed in the present thesis. Even the application of several domains (e.g., component dependencies based on functional and organizational linkages) can be mathematically combined, but must be checked for informational relevance. Such a combination of several domains would permit users to notably enlarge views on a complex system with specific problems.

The enlargement of methodical deduction from DSMs to DMMs is obvious and has already been realized for some specific applications. For example, connectivity maps (see Section 2.3.2) make use of such computations, but so far are not integrated in the comprehensive MDM approach. Whereas the mathematical determination of DMMs and the integration in the MDM approach seems to be unproblematic, the applicability of known analyses, interpretations and possibilities of optimization of DSM methodology are still in question. Due to of the lack of appropriate knowledge, the multitude of analysis criteria applied to DSM structures can not be used for DMM structures (e.g., feedback loops). Only if relevant analyses become available, will these computations provide notable benefit.

The actual development status of the MDM approach already provides possibilities to superpose different sub-matrices (comprising identical nodes but different dependency meaning). De facto, this means a creation of a three-dimensional matrix, which has only been applied to systematic data acquisition and data storage so far. The mathematical combination of several superposed DSMs, i.e., the combination of different dependency meanings within the same nodes, has not been considered in the approach presented. Structural optimization due to multiple criteria could result in information of high relevance, as, for example, the creation of modules due to multiple criteria could be systematized. This enhancement would allow optimizing the organizational structure, for example, by simultaneously taking into account people's connectivity due to component dependencies as well as document flow. Thus, the different system views derived from the Multiple-Domain Matrix could be introduced to a comprehensive optimization approach.

Available MDM computations could also be enhanced by the integration of logical operators, such as mean or extreme values. Although the technical realization appears not to create any problems, the relevance of resulting dependency networks, possible analyses, and the comprehensibility for users has to be discussed.

If partly available quantification could be applied to the MDM approach, the application area could be significantly increased. The procedure presented is completely restricted to non-quantified data because overall data quantifications (that are often requested for the application of simulations) are generally not available in complex product development networks. If existing information on quantifications can be applied to create a Multiple-Domain Matrix, even if further system areas remain non-quantified, that data could be productively applied to development situations. Sub areas of complex networks that are known in detail could be analyzed more closely.

Finally, methodical analysis and interpretation as well as structural optimization provide potential for future research. Although the present thesis provides a large set of structure criteria for analyses and subsequent interpretations, the underlying area of graph theory has

not been completely captured. Besides the structural criteria neglected so far (e.g., colorability or planarity of graphs), the interpretation of criteria combinations is also significant for the management of complex networks in product development. If possibilities of interpretation and rating are on hand, optimization approaches can be improved, as reliable indexes for characterizing complexity become possible. Only a few index values have been developed for the automatic optimization by genetic algorithms, in particular, that can be applied as fitness values for structure validation. Rating possibilities for user-defined criteria (or combinations of criteria) would provide significant potential for future structure-oriented product development.

To sum up, the Multiple-Domain Matrix approach combines relevant groundwork for managing complex system structures to allow for a comprehensive, domain-spanning analysis, control and optimization of structural networks. The methods developed and the related proceedings offer new insights into the structures examined and enable a systematic approach to numerous relevant challenges in science and industry.

Nevertheless, the degree of novelty of the methodology presented suggests numerous improvements, some of which have already been mentioned in this chapter. The promising results and achievements of the current application offer exciting possibilities for future development and research in this area.

6. References

AKAO, Y. (1992):
QFD - Quality Function Deployment.
Landsberg a. L.: Moderne Industrie 1992.

ALEXANDER, C. (1964):
Notes on the Synthesis of Form.
Cambridge: Harvard University Press 1964.

ALLEN, T. T. (2006):
Introduction to Engineering Statistics and Six Sigma – Statistical Quality Control and Design of Experiments and Systems.
London: Springer 2006.

ALTSCHULLER, G. S. (1984):
Erfinden – Wege zur Lösung technischer Probleme.
Berlin: Verlag Technik 1984.

AMBROSY, S. (1996):
Methoden und Werkzeuge für die integrierte Produktentwicklung.
Aachen: Shaker 1997. (Konstruktionstechnik München, Band 26)
Also München: TU, Diss. 1996.

ANDERSON, C. (2006):
The Long Tail: How Endless Choice is Creating Unlimited Demand.
London: Random House Books 2006.

ANDRÁSFAI, B. (1991):
Graph Theory: Flows, Matrices.
Bristol: Adam Hilger 1991.

ANDREASEN, M.; KÄHLER, S.; LUND, T. (1983):
Design for Assembly.
Berlin: Springer 1983.

ASHBY, R. W. (1956):
An Introduction to Cybernetics.
London: Chapman & Hall 1956.

BALDWIN, C. Y.; CLARK, K. B. (2000):
Design Rules - The Power of Modularity. Vol. 1,
Cambridge: MIT Press 2000.

BAUMBERGER, C.; BRAUN, T.; LINDEMANN, U.; MAURER, M. (2006):
Strategic Diversification by Network Portfolio Analysis. In: Marjanovic, D. (Ed.): Proceedings of the 9th International Design Conference 2006 (DESIGN06), Dubrovnik.
Glasgow: Design Society 2006, pp 177-184.

BLANCHARD, B. S. (2004):
Systems Engineering Management. 3rd Ed.
Hoboken: John Wiley & Sons 2004.

BLESSING, L. T. M.; CHAKRABARTI, A.; WALLACE, K. M. (1998):
An Overview of Descriptive Studies in Relation to a General Design Research Methodology. In: Frankenberger, E.; Badke-Schaub, P.; Birkhofer, H. (Eds.): Designers – The Key to Successful Product Development.
London: Springer 1998, pp 42-56.

BOARDMAN, J.; SAUSER, B. (2006):
System of Systems – the Meaning of of. In: Proceedings of the 2006 IEEE/SMC International Conference of Systems Engineering.
Los Angeles: IEEE 2006.

BOLLOBÁS, B. (1990):
Graph Theory.
New York: Springer 1990.

BONGULIELMI, L.; HENSELER, P., PULS, C.; MEIER, M. (2001):
The K- & V-Matrix Method – An Approach in Analysis and Description of Variant Products. In: Proceedings of the 13th International Conference on Engineering Design (ICED 01), Glasgow.
Bury St. Edmunds: IMechE 2001, pp 571-578.

BRABAZON, T.; MATTHEWS, R. (2002):
Product Architecture, Modularity and Product Design: A Complexity Perspective. In: Proceedings of the 2002 British Academy of Management Annual Conference.
Hammersmith: 2002.

BRADY, T. K. (2002):
Utilization of Dependency Structure Matrix Analysis to Assess Complex Project Designs. In: Proceedings of DETC´02: ASME 2002 Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Montréal.
Montréal: ASME 2002.

BROWNING, T. (2001):
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, T. R.; EPPINGER, S. (2002):
Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development. IEEE Transaction on Engineering Management 49 (2002) 4, pp 428–442.

BROY, M. (1998):
Informatik – Systemstrukturen und theoretische Informatik 2. 2nd Ed.,
Berlin: Springer 1998.

BRUALDI, R. A.; RYSER, H. J. (1991):
Combinatorial Matrix Theory.
Cambridge: Cambridge University Press 1991.

BRUEGGE, B.; DUTOIT, A. H. (2000):
Object-Oriented Software Engineering, Conquering Complex and Changing Systems.
Upper Saddle River: Prentice-Hall 2000.

BULLINGER, H.-J., KISS-PREUßINGER, E.; SPATH, D. (EDS.) (2003):
Automobilentwicklung in Deutschland – wie sicher in die Zukunft? Chancen, Potenziale und Handlungsempfehlungen für 30 Prozent mehr Effizienz.
Stuttgart: Fraunhofer-IRB 2003.

BUZAN, T.; BUZAN, B. (2002):
Das Mind-Map-Buch: Die beste Methode zur Steigerung Ihres geistigen Potenzials. 5th Ed.,
Landsberg: mvg 2002.

CHARTRAND, G.; OELLERMANN, O. R. (1993):
Applied and Algorithmic Graph Theory.
New York: McGraw-Hill 1993.

CLARK, K. B.; FUJIMOTO, T. (1991):
Product Development Performance: Strategy, Organization, and Management in the World Auto Industry.
Boston: Harvard Business School Press 1991.

CLARKSON, P. J.; MELO, A.; CONNOR, A. (2000):
Signposting for Design Process Improvement. In: Gero, J. S. (Ed.): Proceedings of the International Conference on Artificial Intelligence in Design (AID 2000), Worcester.
Cambridge: Worcester Polytechnic Institute 2000, pp 333-353.

CLARKSON, P. J.; SIMONS, C.; ECKERT, C. (2001):
Predicting Change Propagation in Complex Design. In: Proceedings of DETC'01: ASME 2001 Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Pittsburgh.
Pittsburgh: ASME 2001.

COYLE, R. G. (1996):
System Dynamics Modelling – A Practical Approach.
London: Chapman & Hall 1996.

CROSS, N. (2000):
Engineering Design Methods: Strategies for Product Design. 3rd Ed.,
New York: John Wiley & Sons 2000.

DAENZER, W. F.; HUBER, F. (1999):
Systems Engineering: Methodik und Praxis. 10th Ed.,
Zürich: Verl. Industrielle Organisation 1999.

DANILOVIC, M.; BÖRJESSON, H. (2001A):
Managing the Multiproject Environment. In: Proceedings of the 3rd Dependence Structure Matrix (DSM) International Workshop, Cambridge.
Cambridge, USA: Massachusetts Institute of Technology 2001.

DANILOVIC, M.; BÖRJESSON, H. (2001B):
Participatory Dependence Structure Matrix Approach. In: Proceedings of the 3rd Dependence Structure Matrix (DSM) International Workshop, Cambridge.
Cambridge, USA: Massachusetts Institute of Technology 2001.

DANILOVIC, M.; SANDKULL, B. (2002):
Managing Complexity and Uncertainty in a Multiproject Environment. In: Proceedings of the 2002 5th International Conference of the International Research Network on Organizing by Projects, Rotterdam.
Rotterdam: Erasmus University 2002.

DANILOVIC, M.; SIGEMYR, T. (2003):
Multiplan – A New Multi-Dimensional DSM-Tool. In: Proceedings of the 5th Dependence Structure Matrix (DSM) International Workshop, Cambridge.
Cambridge, UK: University of Cambridge 2003.

DANILOVIC, M.; BROWNING, T. (2004):
A Formal Approach for Domain Mapping Matrices (DMM) to Complement Design Structure Matrices (DSM). In: Proceedings of the 6th Design Structure Matrix (DSM) International Workshop, Cambridge.
Cambridge, UK: University of Cambridge, Engineering Design Centre 2004.

DEUBZER, F.; KREIMEYER, M.; JUNIOR, T.; ROCK, B. (2005):
Der Änderungsmanagement Report 2005.
Cidad Working Paper Series 1 (2005) 1, pp 1-12.

DI BATTISTA, G.; EADES, P.; TAMASSIA, R.; TOLLIS, I. G. (1999):
Graph Drawing: Algorithms for the Visualization of Graphs.
Upper Saddle River, New Jersey: Prentice Hall 1999.

DICK, B.; LINDEMANN, U. (2006):
Adaptation of Methods to Crises Situations in Product Development. In: Proceedings of NordDesign 2006, Reykjavik.
Reykjavik: Design Society 2006, pp 56-63.

DONG, Q.; WHITNEY, D. (2001):
Designing a Requirement Driven Product Development Process. In: Proceedings of the DETC 2001: ASME 2001 International Design Engineering Technical Conferences. 13th International Conference on Design Theory and Methodology, Pittsburgh.
Pittsburgh: ASME 2001.

DÖRNER, D. (1992):
Die Logik des Misslingens.
Reinbek: Rowohlt Taschenbuch Verlag 1992.

DSM INTERNET PAGE (2007):
Identifying Loops by Powers of the Adjacency Matrix.
[Taken: 26.02.2007, URL:
http://www.dsmweb.org/index.php?option=com_content&task=view&id=35&Itemid=26].

EHRLENSPIEL, K. (1995):
Integrierte Produktentwicklung – Methoden für Prozessorganisation, Produkterstellung und Konstruktion.
München: Hanser 1995.

EICHINGER, M.; MAURER, M.; PULM, U.; LINDEMANN, U. (2006):
Extending Design Structure Matrices and Domain Mapping Matrices by Multiple Design Structure Matrices. In: Proceedings of the 8th Biennial Conference on Engineering Systems Design and Analysis (ASME-ESDA06), Torino.
Torino, Italy: ASME 2006.

EL-HAIK, B.; YANG, K. (1999):
The Components of Complexity in Engineering Design.
IIE Transactions 31 (1999) 10, pp 925-934.

EPPINGER, S. D. (1991):
Model-based Approaches to Managing Concurrent Engineering.
Journal of Engineering Design (1991) 2, pp 283–290.

EPPINGER, S. D.; WHITNEY, D. E.; SMITH, R. P.; GEBALA, D. A. (1994):
A Model-based Method for Organizing Tasks in Product Development.
Res. in Eng. Design (1994) 6, pp 1–13.

EPPINGER, S. D.; SALMINEN, V. (2001):
Patterns of Product Development Interactions. In: Proceedings of the 13th International Conference on Engineering Design (ICED 01), Glasgow.
Bury St. Edmunds: IMechE 2001.

ERICSSON, A.; ERIXON, G. (1999):
Controlling Design Variants – Modular Product Platforms.
New York: ASME Press 1999.

EVEN, S. (1979):
Graph Algorithms.
Potomac: Computer Science Press 1979.

FAYYAD, U.; GRINSTEIN, G.; WIERSE, A. (2002):
Information Visualization in Data Mining and Knowledge Discovery.
San Diego: Academic Press 2002.

FERNANDEZ, C. I. G. (1998):
Integration Analysis of Product Architecture to Support Effective Team Co-Location.
Cambridge: Massachusetts Institute of Technology, Master Thesis 1998.

FERNANDO, E. P. C. (1969):
Use of Interdependency Matrix for Expediting Implementation of an Integrated Development Programme in a Developing Country. In: 2nd International Congress on Project Planning by Network Analysis.
Amsterdam: 1969, pp 76–85.

FIRCHAU, N. L. (2003):
Variantenoptimierende Produktgestaltung.
Göttingen: Cuvillier 2003.

FLANAGAN, T. L.; ECKERT, C. M.; SMITH, J.; EGER, T.; CLARKSON, P. J. (2003):
A Functional Analysis of Change Propagation. In: Proceedings of the 14th International Conference on Engineering Design (ICED 03), Stockholm.
Stockholm: Design Society 2003.

FORRESTER, J. W. (1961):
Industrial Dynamics.
Cambridge: MIT Press 1961.

FORRESTER, J. W. (1980):
System Dynamics – Future Opportunities. In: Legasto, A. A.; Forrester, J. W.; Lyneis, J. M. (Eds.): System Dynamics: Studies in the Management Sciences.
Amsterdam: North-Holland Publishing Company 1980, pp 7-21.

FOULDS, L. R. (1992):
Graph Theory Applications.
New York: Springer 1992.

FRANKE, H.-J.; HESSELBACH, J.; FIRCHAU, N. L.; HUCH, B. (2002):
Variantenmanagement in der Einzel- und Kleinserienfertigung.
München: Hanser 2002.

FREEMAN, L. C. (1979):
Centrality in Social Networks: Conceptual Clarification.
Social Networks 1 (1978/1979), pp 215-239.

GAUSEMEIER, J.; FINK, A.; SCHLAKE, O. (1995):
Szenario-Management: Planen und Führen mit Szenarien.
München: Hanser 1995.

GEBALA, D. A.; EPPINGER, S. D. (1991):
Methods for Analyzing Design Procedures. In: Proceedings of the ASME Third International Conference in Design Theory and Methodology, Miami.
Miami, USA: ASME 1991, pp 227-233.

GOLDBERG, D. E. (1989):
Genetic Algorithms in Search, Optimization, and Machine Learning.
Reading: Addison-Wesley 1989.

GOMEZ, P.; PROBST, G. (1997):
Die Praxis des ganzheitlichen Problemlösens. 2nd Ed.,
Bern: Paul Haupt 1997.

GROSE, D. L. (1994):
Reengineering the Aircraft Design Process. In: Proceedings of the Fifth AIAA/USA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City Beach.
Panama City Beach, USA: NASA 1994.

GROSS, J. L.; YELLEN, J. (2006):
Graph Theory and its Applications. 2nd Ed.,
Boca Raton: Chapman & Hall/CRC 2006.

HANGOS, K. M.; CAMERON, I. T. (2001):
Process Modelling and Model Analysis.
San Diego: Academic Press 2001.

HARASHIMA, F.; TOMIZUKA, M.; FUKUDA, T. (1996):
Mechatronics – What Is It, Why and How? An Editorial.
IEEE/ASME Transactions on Mechatronics 1 (1996) 1, pp 1-4.

HARTIGAN, J. A. (1975):
Clustering Algorithms.
New York: John Wiley & Sons 1975.

HAUSER, J. R.; CLAUSING, D. (1988):
The House of Quality.
Harvard Business Review 66 (1988) 3, pp 63-73.

HAYES, M. (1969):
The Role of Activity Precedence Relationships in Node-oriented Networks. In: 2nd International Congress for Project Planning by Network Analysis.
Amsterdam: 1969, pp 128–146.

HERDER-DORNEICH, P. (1992):
Vernetzte Strukturen – Das Denken in Ordnungen.
Baden-Baden: Nomos Verlagsgesellschaft 1992.

HUB, H. (1994):
Ganzheitliches Denken im Management: Komplexe Aufgaben PC-gestützt lösen.
Wiesbaden: Gabler 1994.

HUBBERT, J. (2003):
Bis an die Grenzen gefordert.
Automobilindustrie 48 (2003) 11, pp. 28-32.

HUBKA, V.; EDER, E. W. (1996):
Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge.
London: Springer 1996.

INCOSE (INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING) (2002):
Systems Engineering Guidebook: A how to Guide for all Engineers. Version 2, 2002,
[Taken 20.06.2007, URL: http://www.incose.org].

JARRATT, T. (2004):
A Model-Based Approach to Support the Management of Engineering Change.
Cambridge (UK): University of Cambridge, Diss. 2004.

JOHNSON, C. (ED.) (2005):
2nd Workshop on Complexity in Design and Engineering.
GIST Technical Report G2005-1,
Glasgow: University of Glasgow, Department of Computing Science 2005.

KARNOPP, D.; MARGOLIS, D.; ROSENBERG, R. (1990):
System Dynamics: A Unified Approach. 2nd Ed.,
New York: John Wiley & Sons 1990.

KEHAT, E.; SHACHAM, M. (1973):
Chemical Process Simulation Programs–2: Partitioning and Tearing of Systems Flowsheets.
Process Technology International 18 (1973), pp 115–118.

KEIJZER, W.; KREIMEYER, M.; SCHACK, R.; LINDEMANN, U.; ZÄH, R. (2006):
Vernetzungsstrukturen in der Digitalen Fabrik – Status, Trends und Empfehlungen.
München: Dr. Hut 2006.

KOSSIAKOFF, A.; SWEET, W. N. (2003):
Systems Engineering: Principles and Practice.
Hoboken: John Wiley and Sons 2003.

KÖSTER, O. (1998):
Komplexitätsmanagement in der Industrie.
Wiesbaden: DUV 1998.

KREIMEYER, M.; HERFELD, U.; DEUBZER, F.; DEQUIDT, C.; LINDEMANN, U. (2006):
Function-driven Product Design in Virtual Teams through Methodical Structuring of Requirements and Components. In: Proceedings of the 8th Biennial Conference on Engineering Systems Design and Analysis (ASME-ESDA06), Torino.
Torino, Italy: ASME 2006.

KRISHNAN, V.; EPPINGER, S. D.; WHITNEY, D. E. (1997):
Simplifying Iterations in Cross-Functional Design Decision Making.
Journal of Mechanical Design 119 (1997) 4, pp 485-493.

KUSIAK, A. (1999):
Engineering Design – Products, Processes and Systems.
San Diego: Academic Press 1999.

KUSIAK, A. (2000):
Computational Intelligence in Design and Manufacturing.
New York: John Wiley & Sons 2000.

KUSIAK, A.; PARK, K. (1990):
Concurrent Engineering: Decomposition and Scheduling of Design Activities.
International Journal of Production Research 28 (1990), pp 1883–1900.

KUSIAK, A.; LARSON, N.; WANG, J. (1994):
Reengineering of Design and Manufacturing Processes.
Computers and Industrial Engineering 26 (1994) 3, pp 521-536.

KUSIAK, A; TANG, C.-Y.; SONG, Z. (2006):
Identification of Modules with an Interface Structure Matrix. Working Paper ISL_04/2006.
Iowa: The University of Iowa, Intelligent Systems Laboratory 2006.

LINDEMANN, U. (ED.) (2004A):
Marktnahe Produktion individualisierter Produkte.
Sonderforschungsbereich 582, Arbeits- und Ergebnisbericht 2001/2 – 2004/1,
München: Technische Universität 2004.

LINDEMANN, U. (ED.) (2004B):
Marktnahe Produktion individualisierter Produkte – Industriekolloquium des Sonderforschungsbereichs 582.
München: Herbert Utz 2004.

LINDEMANN, U. (2007):
Methodische Entwicklung technischer Produkte. 2nd Ed.,
Berlin: Springer 2007.

LINDEMANN, U.; KLEEDÖRFER, R.; GERST, M. (1998):
The Development Department and Engineering Change Management. In: Frankenberger, E.; Badke-Schaub, P.; Birkhofer, H. (Eds.): Designers: The Key to Successful Product Development.
London: Springer 1998, pp 169-182.

LINDEMANN, U.; MAURER, M.; KREIMEYER, M. (2005):
Intelligent Strategies for Structuring Products. In: Clarkson, J.; Huhtala, M. (Eds.): Engineering Design – Theory and Practice.
Cambridge, UK: Engineering Design Centre 2005, pp 106-115.

LINDEMANN, U.; MAURER, M. (2006):
Early Evaluation of Product Properties for Individualised Products.
International Journal of Mass Customization 1 (2006) 2/3, pp 299-314.

LINDEMANN, U.; REICHWALD, R.; ZÄH, M. F. (2006):
Individualisierte Produkte – Komplexität beherrschen in Entwicklung und Produktion.
Berlin: Springer 2006.

MAIER, A.; KREIMEYER, M.; HERFELD, U.; DEUBZER, F.; LINDEMANN, U.; CLARKSON, P. J. (2006):
Reflecting Communication: A Key Factor for Successful Collaboration between Embodiment Design and Simulation. In: Marjanovic, D. (Ed.): Proceedings of the 9th International Design Conference 2006 (DESIGN06), Dubrovnik.
Glasgow: Design Society 2006.

MALIK, F. (2003):
Strategie des Managements komplexer Systeme.
Bern: Haupt 2003.

MALMQVIST, J. (2002):
A Classification of Matrix-Based Methods for Product Modeling. In: Marjanovic, D. (Ed.): Proceedings of the 7th International Design Conference 2002 (DESIGN02), Cavtat-Dubrovnik. Cavtat-Dubrovnik, Croatia: Design Society 2002.

MAURER, M.; PULM, U.; LINDEMANN, U. (2004):
Utilization of Graph Constellations for the Development of Customizable Product Spectra. In: Fourth International ICSC Symposium on Engineering of Intelligent Systems EIS 2004, Madeira.
Madeira, Portugal: ICSC Interdisciplinary Research Canada 2004.

MAURER, M.; BOESCH, N.-O.; SHENG, G.; TZONEV, B. (2005):
A Tool for Modelling Flexible Product Structures – MOFLEPS. In: Proceedings of the 15th International Conference on Engineering Design (ICED 05), Melbourne.
Melbourne: Institution of Engineers 2005.

MAURER, M.; PULM, U.; BALLESTREM, F.; CLARKSON, J.; LINDEMANN, U. (2006):
The Subjective Aspects of Design Structure Matrices – Analysis of Comprehension and Application and Means to Overcome Differences. In: Proceedings of the 8th Biennial Conference on Engineering Systems Design and Analysis (ASME-ESDA06), Torino.
Torino, Italy: ASME 2006.

MAURER, M.; LINDEMANN, U. (2007):
Facing Multi-Domain Complexity in Product Development.
Cidad Working Paper Series 3 (2007) 1, pp 1-12.

MCCORMICK, J. W. T.; SCHWEIZER, P. J.; WHITE, T. W. (1972):
Problem Decomposition and Data Reorganization by a Clustering Technique.
Operations Research 20 (1972) 5, pp 993-1009.

MELNIKOV, O.; TYSHKEVICH, R.; YEMELICHEV, V.; SARVANOV, V. (1994):
Lectures on Graph Theory.
Mannheim: BI Wissenschaftsverlag 1994.

MIETTINEN, K. (1999):
Evolutionary Algorithms in Engineering and Computer Science.
Chichester: Wiley 1999.

MINNEMANN, S. (1991):
The Social Construction of a Technical Reality – Empirical Studies of Group Engineering Practice.
Palo Alto: Standford University, Ph. D. Thesis 1991.

NASA (NATIONAL AERONAUTICS AND SPACE ADMINISTRATION) (1995):
NASA Systems Engineering Handbook. SP-6105,
http://ldcm.nasa.gov/library/Systems_Engineering_Handbook.pdf,
Taken: 14th May 2007.

PAHL, G.; BEITZ, W. (1996):
Engineering Design: A Systematic Approach. 2nd Ed.,
London: Springer 1996.

PALM, W. (2005):
System Dynamics.
Boston: McGraw-Hill 2005.

PILLER, F. T. (2001):
Mass Customization: Ein wettbewerbsstrategisches Konzept im Informationszeitalter. 2nd Ed.,
Wiesbaden: Gabler 2001.

PILLER, F. T.; WARINGER, D. (1999):
Modularisierung in der Automobilindustrie – neue Formen und Prinzipien. Modular Sourcing, Plattformkonzept und Fertigungssegmentierung als Mittel des Komplexitätsmanagements.
Aachen: Shaker 1999.

PILLER, F. T.; STOTKO, C. M. (2003):
Mass Customization und Kundenintegration – Neue Wege zum innovativen Produkt.
Düsseldorf: Symposion 2003.

PIMMLER, T. U.; EPPINGER, S. D. (1994):
Integration Analysis of Product Decompositions. In: Proceedings of the 1994 ASME Design Theory and Methodology Conference, Minneapolis.
Minneapolis, USA: ASME 1994.

PINE, J. B. II (1993):
Mass Customization: The New Frontier in Business Competition.
Boston: Harvard Business School Press 1993.

PLEGMATUM INTERNET PAGE (2007):
Presentation of the research group Plegmatum.
[Taken: 14.05.2007, URL: http://www.plegmatum.de/].

PONN, J.; LINDEMANN, U. (2005):
Characterization of Design Situations and Processes and a Process Module Set for Product Development. In: Proceedings of the 15th International Conference on Engineering Design (ICED 05), Melbourne.
Melbourne: Institution of Engineers 2005.

PONN, J.; LINDEMANN, U. (2006):
Intelligent Search for Product Development Information – An Ontology-Based Approach. In: Marjanovic, D. (Ed.): Proceedings of the 9th International Design Conference 2006 (DESIGN06), Dubrovnik.
Glasgow: Design Society 2006, pp 1203-1210.

PROBST, G.; GOMEZ, P. (1991):
Vernetztes Denken: Ganzheitliches Führen in der Praxis.
Wiesbaden: Gabler 1991.

PUHL, H. (1999):
Komplexitätsmanagement.
Kaiserslautern: Univ. Kaiserslautern, Diss. 1999.

PULM, U. (2004):
Eine systemtheoretische Betrachtung der Produktentwicklung.
München: Dr. Hut 2004. (Produktentwicklung München, Band 56)
Also München: TU, Diss. 2004.

PULS, C. (2002):
Die Konfigurations- & Verträglichkeitsmatrix als Beitrag zum Management von Konfigurationswissen in KMU.
Zürich: ETH, Diss. 2002.

PULS, C.; BONGULIELMI, L.; HENSELER, P.; MEIER, M. (2002):
Management of Different Types of Configuration Knowledge with the K- & V-Matrix and Wiki. In: Marjanovic, D. (Ed.): Proceedings of the 7th International Design Conference 2002 (DESIGN02), Cavtat-Dubrovnik.
Cavtat-Dubrovnik, Croatia: Design Society 2002.

RAPP, T. (1999):
Produktstrukturierung.
Wiesbaden: Gabler 1999.
Also St. Gallen: Univ. St. Gallen, Diss. 1999.

RECHENBERG, I. (1973):
Evolutionsstrategie.
Stuttgart: Friedrich Frommann 1973.

REINHART, G.; LINDEMANN, U.; HEINZL, J.(1996):
Qualitätsmanagement.
Berlin: Springer 1996.

RENNER, I. (2007):
Methodische Unterstützung funktionsorientierter Baukastenentwicklung am Beispiel Automobil.
München: TU, Diss. 2007. (under publication)

RIEDL, R. (2000):
Strukturen der Komplexität – Eine Morphologie des Erkennens und Erklärens.
Berlin: Springer 2000.

SABBAGHIAN, N.; EPPINGER, S.; MURMAN, E. (1998):
Product Development Process Capture & Display Using Web-based Technologies. In: 1998 IEEE International Conference on Systems, Man, Cybernetics, San Diego.
San Diego, USA: IEEE 1998, pp 2664–2669.

SCHUH, G.; SCHWENK, U. (2001):
Produktkomplexität managen.
München: Hanser 2001.

SCHWEFEL, H.-P. (1995):
Evolution and Optimum Seeking.
New York: Wiley & Sons 1995.

SHARMAN, D. M.; YASSINE, A. (2004):
Characterizing Complex Product Architectures.
Systems Engineering 7 (2004) 1, pp 35-60.

SOSA, M. (2005):
A Network Approach to Define Component Modularity. In: Proceedings of the 7th International Dependency Structure Matrix (DSM) Conference, Seattle.
Seattle, USA: Boeing 2005.

SOSA, M. E.; EPPINGER, S. D.; ROWLES, C. M. (2003):
Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions.
Journal of Mechanical Design 125 (2003) 2, pp 240-252.

SOUKUP, T.; DAVIDSON, I. (2002):
Visual Data Mining: Techniques and Tools for Data Visualization and Mining.
New York: Wiley Computer Publishing 2002.

STEINMEIER, E. (1999):
Realisierung eines systemtechnischen Produktmodells – Einsatz in der PKW-Entwicklung.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 28)
Also München: TU, Diss. 1998.

STETTER, R. (2000):
Method Implementation in Integrated Product Development.
München: Dr. Hut 2000. (Produktentwicklung München, Band 41)
Also München: TU, Diss. 2000.

STEWARD, D. (1962):
On an Approach to the Analysis of the Structure of Large Systems of Equations.
SIAM Review 5 (1962), pp 321–342.

STEWARD, D. (1981):
The Design Structure System: A Method for Managing the Design of Complex Systems.
IEEE Transaction on Engineering Management 28 (1981) 3, pp 79-83.

STRONG, M. B. (2002):
Tools and Metrics for Evaluating Modular Product Concepts Based on Strategic Objectives.
Brigham: Brigham Young University, Department of Mechanical Engineering, Master Thesis 2002.

SUH, N. P. (1988):
The Principles of Design.
Oxford: Oxford University Press 1988.

TERNINKO, J.; ZUSMAN, A.; ZLOTIN, B. (1998):
Systematic Innovation: An Introduction to TRIZ (Theory of Inventive Problem Solving).
Boca Raton: St. Lucie Press 1998.

THEBEAU, R. E. (2001):
Knowledge Management of System Interfaces and Interactions for Product Development Processes.
Cambridge: Massachusetts Institute of Technology, System Design & Management Program, Master Thesis 2001.

THULASIRAMAN, K.; SWAMY, M. (1992):
Graphs: Theory and Algorithms.
New York: John Wiley & Sons 1992.

TOUCHGRAPH INTERNET PAGE (2007):
TouchGraph open source components.
[Taken: 27.02.2007, URL: http://www.touchgraph.com/].

TSIEN, H. S. (1955):
Engineering Cybernetics.
London: McGraw-Hill 1955.

TUFAST INTERNET PAGE (2007):
Description of the Student Race Car Project.
[Taken: 10.04.2007, URL: http://www.tufast.de/].

ULRICH, K.; EPPINGER, S. (1995):
Product Design and Development.
New York: McGraw-Hill 1995.

ULRICH, H.; PROBST, G. (2001):
Anleitung zum ganzheitlichen Denken und Handeln – Ein Brevier für Führungskräfte.
Bern: Paul Haupt 2001.

USHER, J. M.; ROY, U.; PARSAEI, H. R. (1998):
Integrated Product and Process Development – Methods, Tools, and Technologies.
New York: John-Wiley & Sons 1998.

VESTER, F. (2000):
Die Kunst vernetzt zu denken.
Stuttgart: DVA 2000.

VESTER, F.; HESLER, A. (1980):
Sensitivitätsmodell.
Frankfurt a. M: Regionale Planungsgemeinschaft Untermain 1980.

WARE, C. (2004):
Information Visualization – Perception for Design. 2nd Ed.
Amsterdam: Elsevier 2004.

WARFIELD, J. N. (1973):
Binary Matrices in System Modeling.
IEEE Transactions on Systems, Man and Cybernetics SMC-3 (1973) 5, pp 441–449.

WASSON, C. S. (2006):
System Analysis, Design, and Development: Concepts, Principles, and Practices.
Hoboken: Wiley-Interscience 2006.

WEBER, C. (2005):
What Is "Complexity"? In: Proceedings of the 15th International Conference on Engineering Design (ICED 05), Melbourne.
Melbourne: Institution of Engineers 2005.

WEINBERG, G. M. (1975):
An Introduction to General Systems Thinking.
New York: Wiley 1975.

WHITEFIELD, R. I.; DUFFY, A. H. B.; ZICHAO, W.; MEECHAN, J. (2001):
Intelligent Design Guidance. In: Proceedings of the 14th International Conference on Engineering Design (ICED 03), Stockholm.
Stockholm: Design Society 2003.

WHITNEY, D. E.; DONG, Q.; JUDSON, J.; MASCOLI, G. (1999):
Introducing Knowledge-based Engineering into an Interconnected Product Development Process. White Paper Jan. 27, 1999.

WIENER, N. (1948):
Cybernetics or Control and Communication in the Animal and the Machine.
New York: Technology Press 1948.

WILDEMANN, H.; ANN, C.; BROY, M.; GÜNTHNER, W. A.; LINDEMANN, U. (2007):
Plagiatschutz – Handlungsspielräume der produzierenden Industrie gegen Produktpiraterie.
München: TCW Transfer-Centrum GmbH & Co. KG 2007.

WINSTON, W. L. (2004):
Operations Research – Applications and Algorithms. 4th Ed.
Belmont: Thomson Brooks/Cole 2004.

WULF, J. (2002):
Elementarmethoden zur Lösungssuche.
München: Dr. Hut 2002. (Produktentwicklung München, Band 49)
Also München: TU, Diss. 2002.

YASSINE, A. (2004):
An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method.
Italian Management Review (2004) 9, [Taken: 01/06/2005, URL: http://www.ge.uiuc.edu/pdlab/Papers/DSM-Tutorial.pdf].

YASSINE, A.; FALKENBURG, D.; CHELST, K. (1999):
Engineering Design Management: An Information Structure Approach.
International Journal of Production Research 37 (1999), pp 2957–2975.

YASSINE, A.; WHITNEY, D.; LAVINE, J.; ZAMBITO, T. (2000):
Do-It-Right-First-Time (DRFT) Approach to Design Structure Matrix (DSM) Restructuring.
In: Proceedings of DETC 2000: ASME 2000 International Design Engineering Technical Conferences, Baltimore.
Baltimore, USA: ASME 2000.

YASSINE, A.; WHITNEY, D. E.; ZAMBITO, T. (2001):
Assessment of Rework Probabilities for Simulating Product Development Processes Using the Design Structure Matrix (DSM). In: Proceedings of DETC 2001: ASME 2001 Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Pittsburgh.
Pittsburgh, USA: ASME 2001.

YASSINE, A.; WHITNEY, D.; DALEIDEN, S.; LAVINE, J. (2003):
Connectivity Maps: Modeling and Analysing Relationships in Product Development Processes. Journal of Engineering Design 14 (2003) 3, pp 377-394.

YU, T.; YASSINE, A.; GOLDBERG, D. E. (2003):
A Genetic Algorithm for Developing Modular Product Architecture. In: Proceedings of DETC 2003: ASME 2003 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago.
Chicago, USA: ASME 2003.

YU, T.; YASSINE, A.; GOLDBERG, D. E. (2005):
An Information Theoretic Method for Developing Modular Architectures Using Genetic Algorithms. IlliGAL Report No. 2005014.
Illinois: University of Illinois at Urbana-Champaign, Department of General Engineering 2005.

ZANKER, W. (1999):
Situative Anpassung und Neukombination von Entwicklungsmethoden.
Aachen: Shaker 1999. (Konstruktionstechnik München, Band 36)
Also München: TU, Diss. 1999.

Theses composed in conjunction with the present work

BALLESTREM, F. (2005):
Individual Comprehension and Application of Design Structure Matrices.
München: TU, Lehrstuhl für Produktentwicklung, Semester Thesis 2005.

EICHINGER, M. (2005):
Multiple Design Structure Matrices – Implementation and Application.
München: TU, Lehrstuhl für Produktentwicklung, Semester Thesis 2005.

KESPER, H. (2006):
Scenario building for Reducing System Complexity by Genetic Algorithms.
München: TU, Lehrstuhl für Produktentwicklung, Semester Thesis 2006.

RUPP, T. (2005):
Evolutionäre Algorithmen für die Komplexitätsverarbeitung in abstrakten Produktspezifikationen.
München: TU, Lehrstuhl für Produktentwicklung, Semester Thesis 2005.

7. Appendix

The Appendix provides an easy access to basic analysis criteria and methods for structure management and optimization presented in this thesis. This chapter can serve as a reference book for the practical application in structure analysis.

For every structural criterion mentioned, a definition is given and the significance is considered depending on the network type. Here, static and dynamic networks are differentiated: Static networks can, for example, represent impact dependencies between product components. Dynamic networks comprise a time component and can, for example, represent the information flow between process tasks. Furthermore, the challenge described has to be tackled in order to achieve a structural improvement. Next, the structural optimization potential is shown, which can be realized by purposeful adaptations to the system elements forming the specific analysis criterion. All basic analysis criteria are clarified by the graph visualization of a structure representing change impacts between the components of an automotive engine. For example, a directed dependency from the “Crankshaft” to the “Chain drive” means that a constructive adaptation to the “Crankshaft” can provoke a subsequent adaptation to the “Chain drive” (depending on the extent of the initial adaptation). For every basic analysis criterion, related literature is mentioned.

The basic analysis criteria are classified into three groups according to the descriptions in section 3.5.3. These groups differ in their objective of structural characterizations. The first group focuses on the characterization of nodes and edges (section 7.1), the second one on structural subsets (section 7.2), and the third one on complete graphs (section 7.3). Within each group the criteria are sorted alphabetically.

In section 7.4 methods for the construction of a structure manual are listed alphabetically. These methods are introduced in section 3.6.1 of the thesis and serve for the improved management of existing structures. The layout of the method descriptions is identical to the one used for the analysis criteria in the sections 7.1 to 7.3. Also the same scenario is used for visual clarification.

In section 7.5, methods for optimizing system structures are listed, which are introduced in section 3.6.2 of the present thesis. Information about the evolutionary algorithm shows the practical application of an optimization process to the structure of a packing machine. This scenario and the specific optimization approach are shown in section 4.1.1, and section 7.5 comprises corresponding computation data.

7.1 Basic analysis criteria characterizing nodes and edges

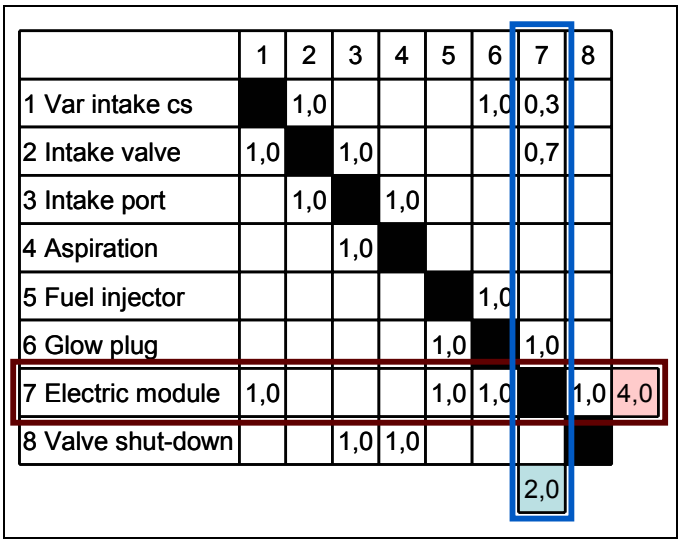
In the following, 13 basic analysis criteria are listed in alphabetical order. These criteria allow single nodes and edges to be characterized in a dependency network. Relevant network elements are highlighted in color in the graphical examples.

7.1.1 Active sum, passive sum

Definition

The active sum indicates the amount (and intensity) of all outgoing edges of a node (highlighted in red in the right figure).

The passive sum indicates the amount (and intensity) of all incoming edges of a node (highlighted in blue in the right figure).



	1	2	3	4	5	6	7	8	
1 Var intake cs		1,0				1,0	0,3		
2 Intake valve	1,0		1,0				0,7		
3 Intake port		1,0		1,0					
4 Aspiration			1,0						
5 Fuel injector						1,0			
6 Glow plug					1,0		1,0		
7 Electric module	1,0				1,0	1,0		1,0	4,0
8 Valve shut-down			1,0	1,0					
							2,0		

Significance in static and dynamic networks

A high (low) value of the active sum means large (little) impact from the concerned node to other nodes in the structure.

A high (low) value of the passive sum means large (little) impact from other nodes in the structure to the node in question.

Challenge of structure improvement

Nodes have to be identified that possess relevant active or passive sums in either the entire system or relevant subsets. Nodes may possess average active or passive sums in the entire system and extremely high or low values in specific subsets. That means that the correct selection of the system part considered is of major importance for obtaining significant analysis results.

Potential

Nodes can be identified which possess active or passive sums that do not suit their desired system role. The focus on relevant nodes allows for the determination of purposeful and effective system changes.

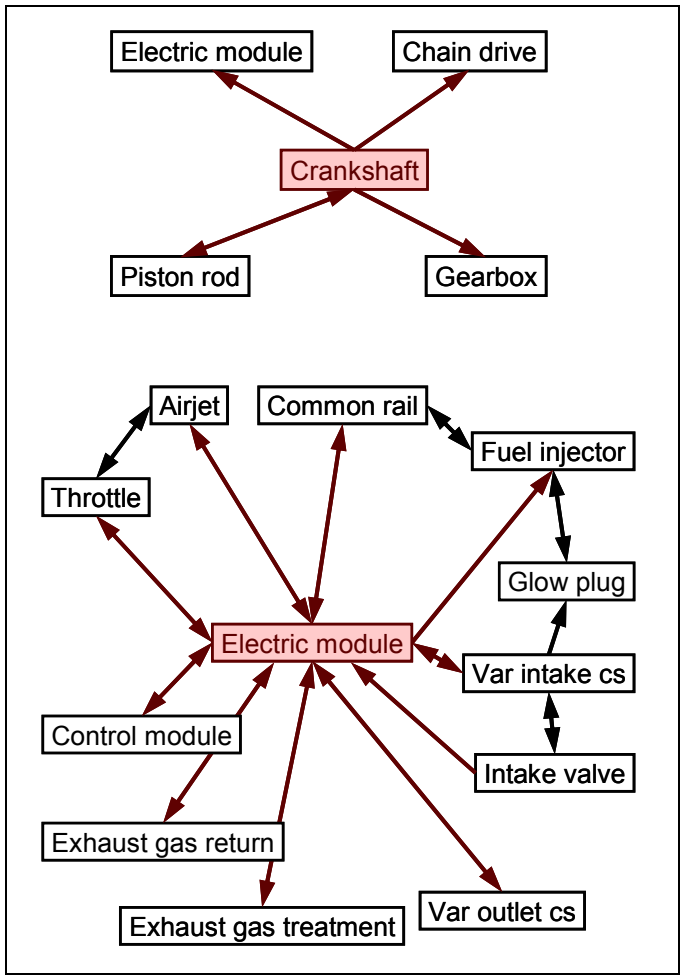
Literature

[DAENZER & HUBER 1999, P. 559], [PROBST & GOMEZ 1991, P. 13F],
[MELNIKOV ET AL. 1994, P. 279FF]

7.1.2 Activity

Definition

The activity of a node is composed by dividing its active sum by its passive sum. The activity is a ratio for comparing different nodes regarding their relative tendency toward an active or passive behavior in a system.



Significance in static and dynamic networks

Nodes with a high activity value possess the tendency to predominantly impact other nodes in the structure, whereas nodes with a low activity value tend are affected by other nodes in the structure.

Challenge of structure improvement

Nodes have to be identified that possess relevant activity values in either the entire system or relevant subsets. Nodes may possess average activity values in the entire system and extremely high or low values in specific subsets. That means that the correct selection of the system part considered is of major importance for obtaining significant analysis results.

Potential

Nodes can be identified, which possess activity values that do not suit their desired system role. The focus on relevant nodes allows for the determination of purposeful and effective system changes.

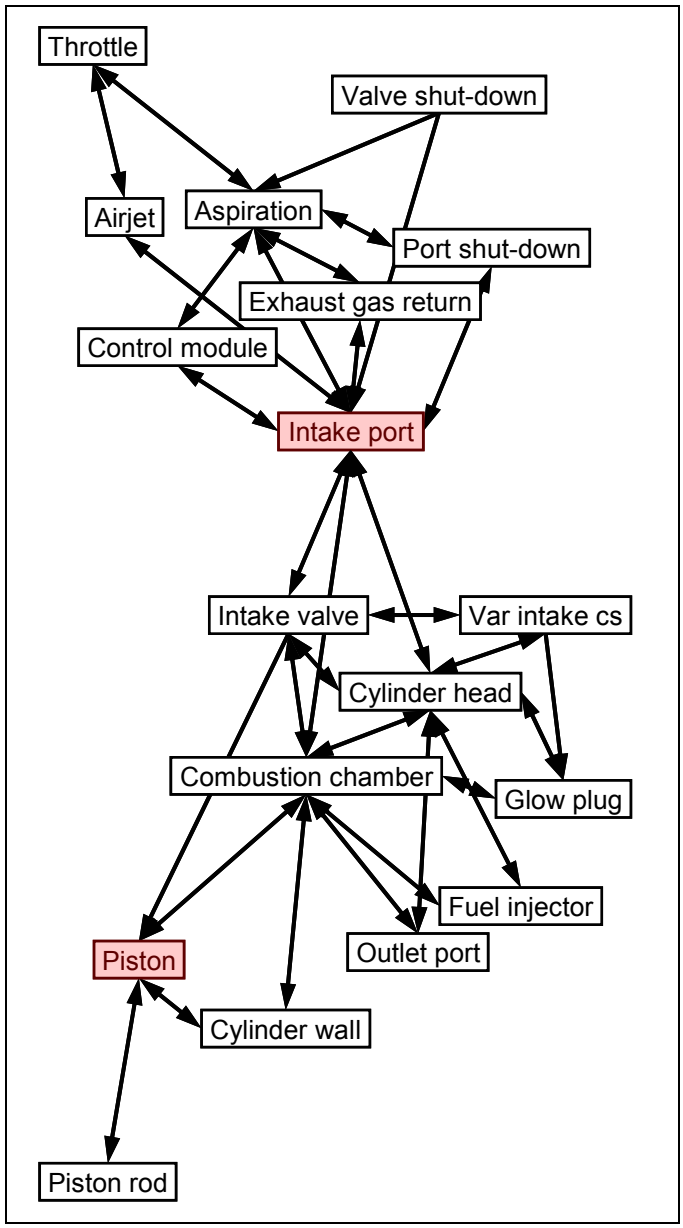
Literature

[PROBST & GOMEZ 1991, P. 13F]

7.1.3 Articulation node

Definition

An articulation node represents the only connection between two subsets of a structure. Impact from one subset to the other one must pass through the articulation node.



Significance in static networks

If the linkage between two subsets is required, one single articulation node between them can be security-sensitive. If the system structure is modularized, an articulation node represents an interface between modules and is suitable for effective decoupling.

Significance in dynamic networks

Articulation nodes form bottle-necks and may restrict transfer volume. Decoupling can prevent any transfer between two subsets.

Challenge of structure improvement

The existence of articulation nodes in a structure depends on the defined system borders. Articulation nodes may occur, if subsets are considered independently from other system parts. In order to interpret articulation nodes correctly it is important to consider a system model that is appropriate for solving the specific problem.

Potential

If relevant articulation nodes can be identified, the connectivity between subsets can be enforced (e.g., by adding further edges) or subsets can be split up into independent modules.

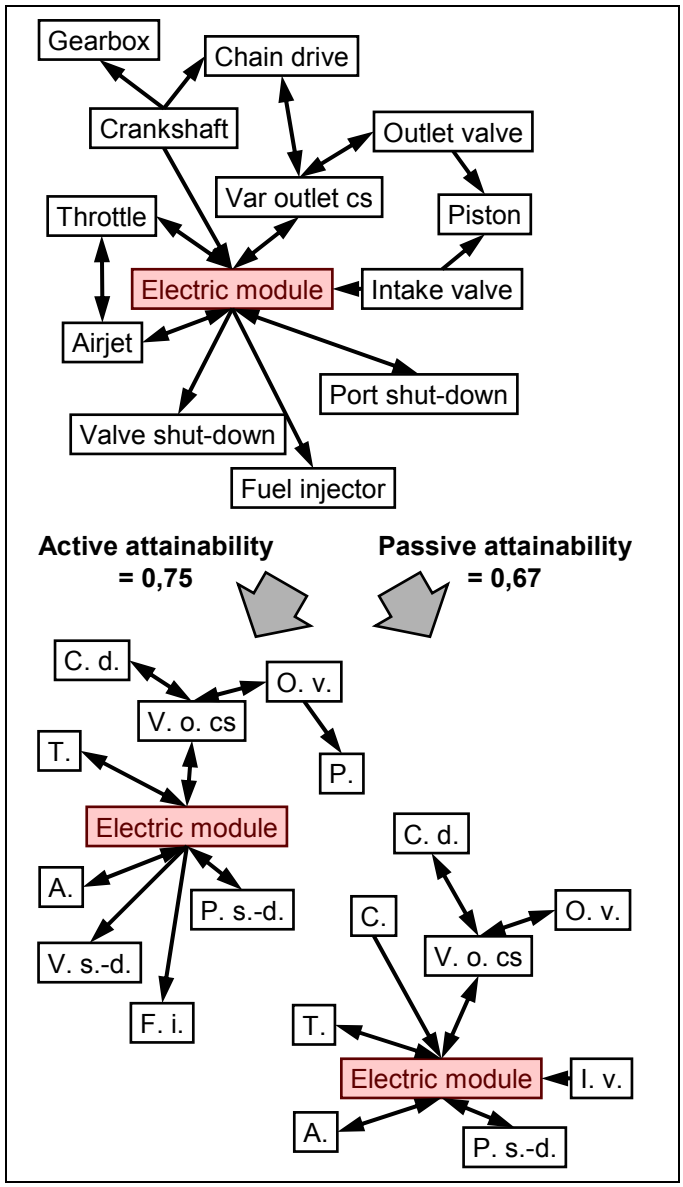
Literature

[CHARTRAND & OELLERMANN 1993, P. 22FF]

7.1.4 Attainability

Definition

The active (passive) attainability of a node is defined as the number of nodes it reaches (it is reached from), divided by the maximum number of nodes it could theoretically reach (it could be reached from), i.e., the number of all other nodes in the system. The term reachability is also used in the literature.



Significance in static networks

A high value of active (passive) attainability means that many other nodes can be affected by any change impact to the considered node (can affect the node due to its adaptation).

Significance in dynamic networks

A high value of active (passive) attainability means that a node is highly (minimally) integrated in the network and possesses high (low) importance.

Challenge of structure improvement

The attainability only provides a vague indication for the structural relevance of a node, as the dependency paths and lengths are not considered. For reliable node characterization, further criteria must be taken into account (e.g., the closeness).

Potential

If it is possible to reduce a node's attainability, the impact to or from other nodes can be reduced in static networks. An increase of attainability means the improvement of a node's integration into the system in dynamic networks.

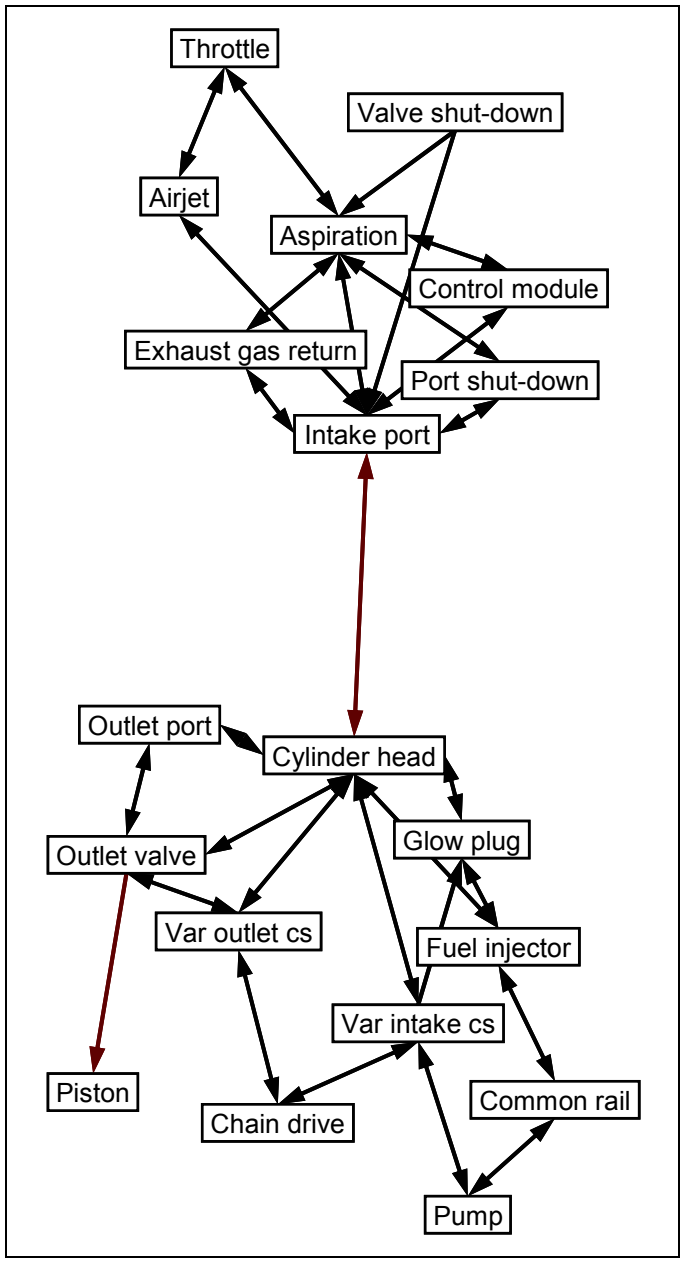
Literature

[FREEMAN 1979], [SOSA 2005]

7.1.5 Bridge edge

Definition

A bridge edge represents the only connection between two subsets of a structure. Impact from one subset to the other one must pass through the bridge edge.



Significance in static networks

If the linkage between subsets is required (e.g., for functionality of the system), one single bridge edge between them can be security-sensitive.

Significance in dynamic networks

Bridge edges form bottle-necks which may restrict transfer volume. Decoupling can prevent any transfer between two subsets.

Challenge of structure improvement

The existence of bridge edges in a structure depends on the defined system borders. Bridge edges may occur, if subsets are considered independently from other system parts. In order to interpret bridge edges correctly, it is important to consider a system model that is appropriate for solving the specific problem.

Potential

If relevant bridge edges can be identified, the connectivity between subsets can be enforced (e.g., by adding edges) or subsets can be split up into independent modules.

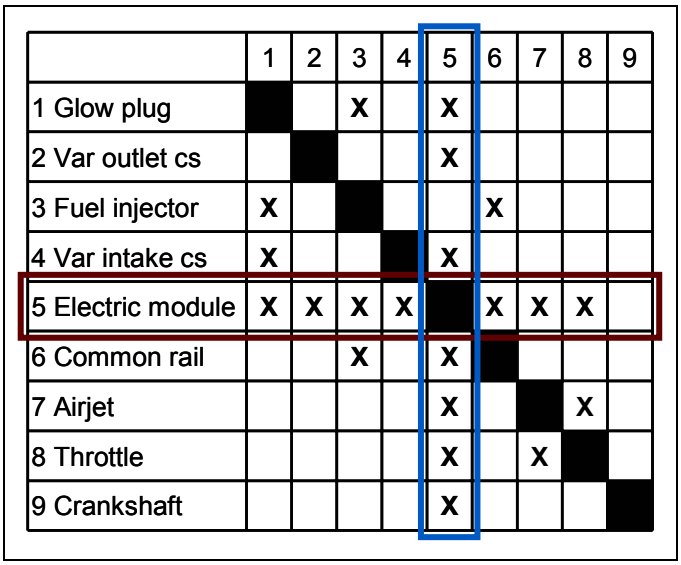
Literature

[CHARTRAND & OELLERMANN 1993, P. 22FF]

7.1.6 Bus

Definition

An active (passive) bus represents a node that possesses many outgoing (incoming) edges.



	1	2	3	4	5	6	7	8	9
1 Glow plug	■		X		X				
2 Var outlet cs		■			X				
3 Fuel injector	X		■			X			
4 Var intake cs	X			■	X				
5 Electric module	X	X	X	X	■	X	X	X	
6 Common rail			X		X	■			
7 Airjet					X		■	X	
8 Throttle					X		X	■	
9 Crankshaft					X				■

Significance in static networks

Changes made to an active bus can result in change impact to many other nodes of the structure. A passive bus will likely be affected by adaptations to user-defined nodes in the structure.

Significance in dynamic networks

An active bus represents an emitter that provides input for a multitude of other nodes. A passive bus represents a receiver that integrates output from many other nodes.

Challenge of structure improvement

Busses can easily be identified in matrix depictions, as rows (active bus) or columns (passive bus) are largely filled. The characterization of a node as a bus depends on the defined system borders. Whereas a node can represent a bus in one specific subset, it may have average numbers of dependencies in the entire structure.

Potential

If active busses are known, their purposeful adaptation allows a large quantity of other nodes to be directly adapted, and therefore the system behavior can be directly adapted, too. Nodes representing passive busses can be designed for enhanced independence from adaptations to other nodes (e.g., by interface standardization or safety dimensioning)

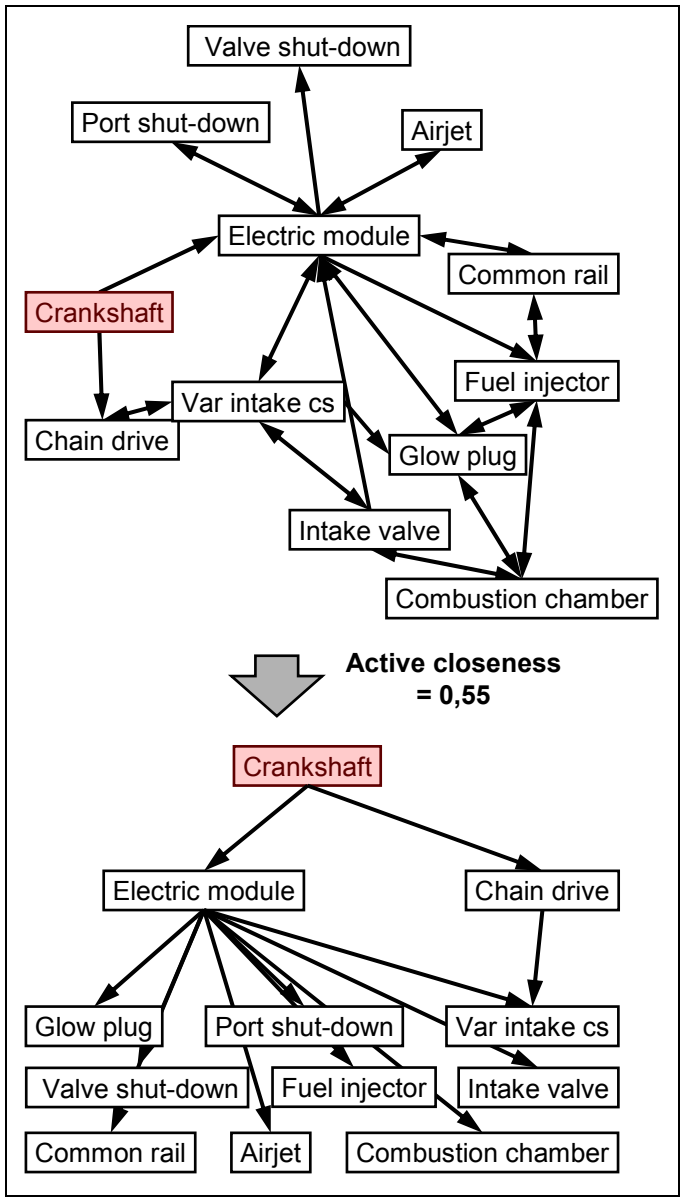
Literature

[YU ET AL. 2003], [SHARMAN & YASSINE 2004]

7.1.7 Closeness

Definition

The active (passive) closeness of a node is defined as the number of nodes that can be reached from the actual node (that can reach the actual node) divided by the overall length of outgoing (incoming) shortest paths to (from) these nodes.



Significance in static networks

A high (low) value of the closeness means that nodes linked to the node considered tend to be connected by short (long) dependency paths. The shorter a connection, the more direct the impact becomes. The longer a path is, the more difficult its detection or control becomes.

Significance in dynamic networks

A high (low) value of the closeness means that a node is centrally integrated in the network. Flows are passed with only a short delay to and from this node.

Challenge of structure improvement

The closeness only provides a vague indication for the structural relevance of nodes. For reliable node characterization, further criteria must be taken into account (e.g., the attainability).

Potential

If it is possible to increase a node's closeness, the impact to/from further system nodes becomes generally more direct and can result in better system control.

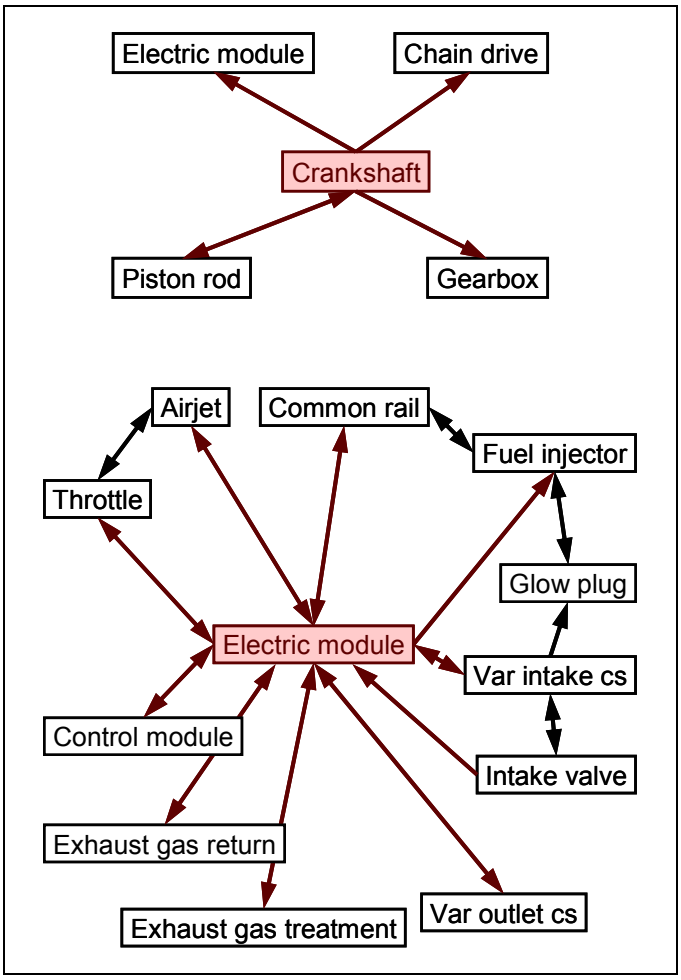
Literature

[FREEMAN 1979]

7.1.8 Criticality

Definition

The criticality of a node is composed by multiplying its active sum with its passive sum. The criticality is a ratio for comparing different nodes regarding their relative sensitivity towards adaptations to a network.



Significance in static and dynamic networks

A high (low) criticality value means that the node will likely (unlikely) be involved in adaptation processes of the system. A node possessing a high criticality can either spread change impact to many other nodes (due to a high active sum), or/and can be affected by other nodes (due to a high passive sum). Medium values for active and passive sums of one node result in a high criticality value as well; in this case the node can influence other nodes and be influenced by other nodes in equal measures.

Challenge of structure improvement

Nodes must be identified that possess relevant criticality values in either the entire system or specific subsets. Nodes may possess average criticality values in the entire system and high or low values in specific subsets. That means that the correct selection of the part considered is of major importance for obtaining significant analysis results.

Potential

Nodes can be identified which possess criticality values that do not suit their desired system role. The focus on relevant nodes allows effective system adaptations to be determined.

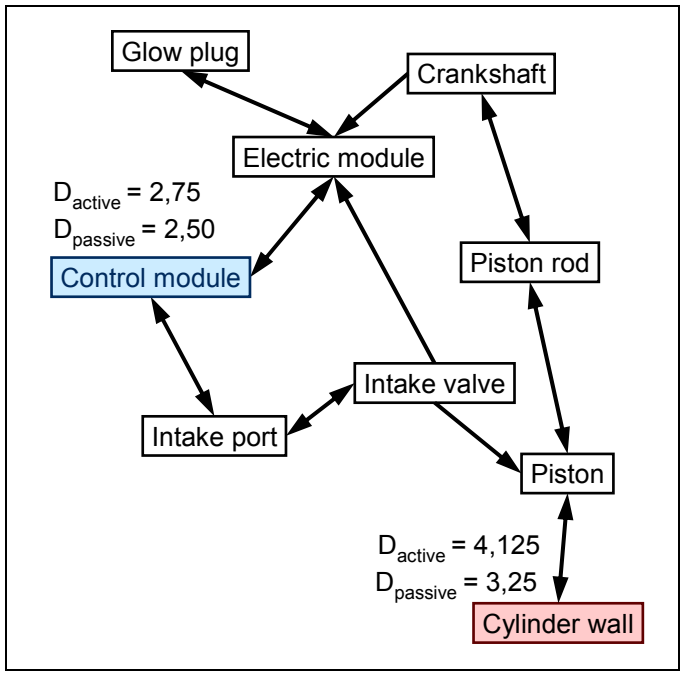
Literature

[PROBST & GOMEZ 1991, P. 13F]

7.1.9 Distance (global)

Definition

The global distance describes the average length of all shortest paths between the characterized node and all directly and indirectly connected nodes of a structure. The active global distance considers the outgoing dependency paths, whereas the passive global distance considers the incoming ones.



Significance in static and dynamic networks

The global distance of a node describes how direct or indirect the node is embedded in relation to its surrounding nodes. A low global distance means that the majority of connected nodes can be reached directly or by short dependency paths. Impacts from or to those nodes generally can spread quickly and without much moderation. A high global distance describes a node that is mostly connected to other nodes by indirect dependency paths. Delay in change propagation and increased moderation effects can occur.

Challenge of structure improvement

Nodes have to be identified that possess relevant values of global distance in either the entire system or specific subsets. Nodes may possess average values of global distance in the entire system and high or low values in specific subsets. This means that the correct selection of the considered part is of major importance for obtaining significant analysis results.

Potential

Nodes can be identified which possess global distance values that do not suit their desired system role. The focus on relevant nodes allows effective system adaptations to be determined.

Literature

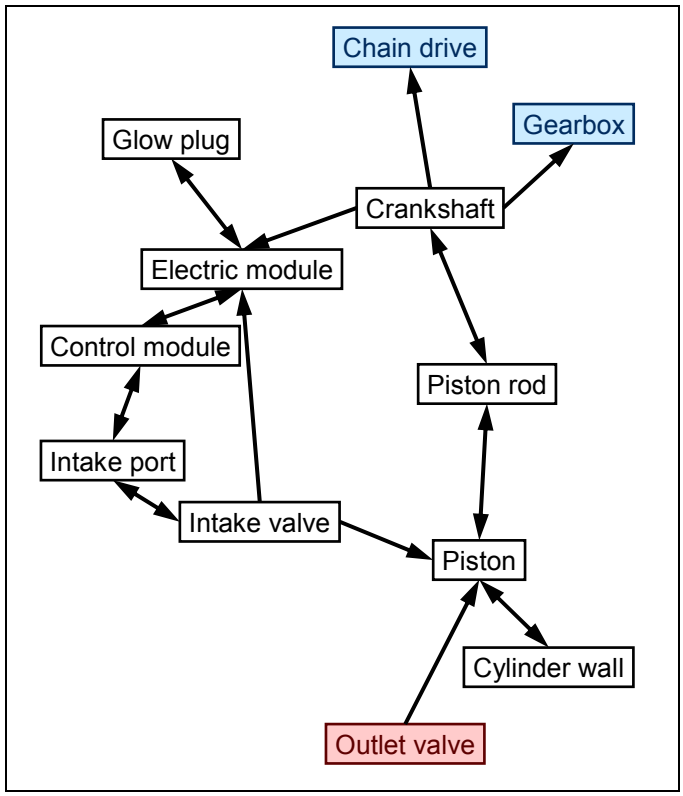
[CHARTRAND & OELLERMANN 1993, P. 99FF], [FREEMAN 1979]

7.1.10 End node, start node

Definition

A start node only possesses outgoing edges to other nodes.

An end node only possesses incoming edges from other nodes.



Significance in static networks

If the dependencies represent change impact, a start node is not affected by adaptations to other nodes of the structure. Adaptations to end nodes can not lead to change impacts to other nodes of the structure.

Significance in dynamic networks

Start nodes are arranged before the nodes they connect to in a process or time line, as subsequent nodes are based on their output. They represent inputs to the system considered. End nodes are located next to nodes they are connected to. These end nodes offer outputs for the system considered.

Challenge of structure improvement

Start and end nodes are easy to detect, for example, in matrix representations (empty columns or rows). They can be useful to review if the practical meaning of identified start and end nodes correspond to the structural significance of input or output parameters (or if further edges must be included in the consideration). Start and end nodes often occur at the chosen system borders; i.e., the existence of start and end nodes often depend on the selected scope of structural considerations.

Potential

Start nodes allow for the manipulation of a structure without the risk of feedback loop effects that reach back to the start node itself (however, other feedback loops that exist in the structure can be activated by adaptations to start nodes). End nodes allow for their adaptation without the risk for change propagation to other nodes of a structure.

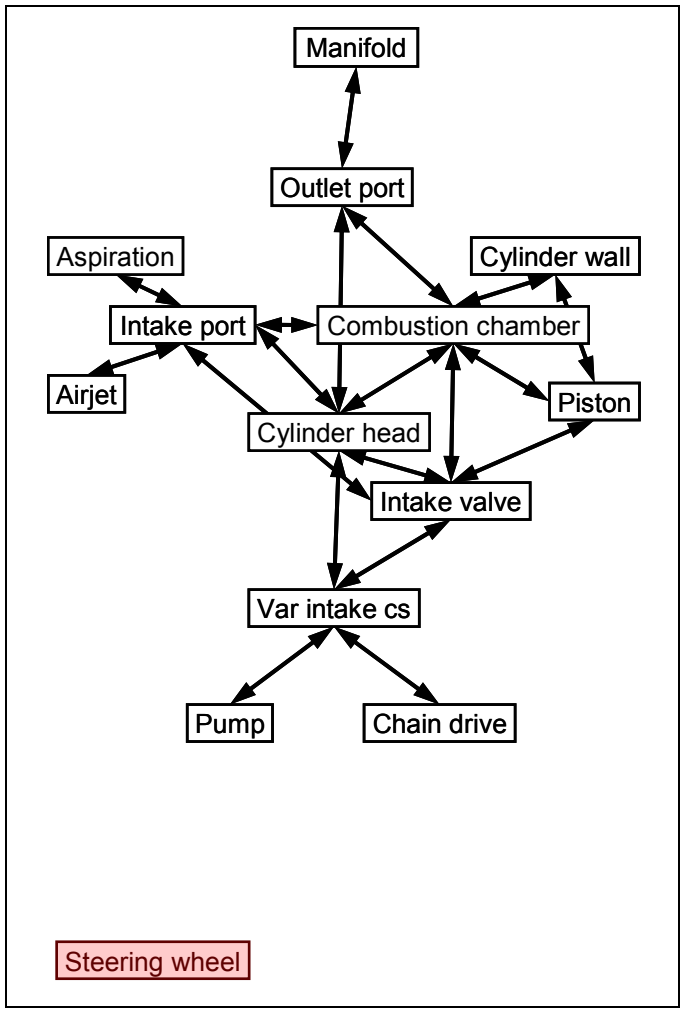
Literature

[BOLLOBÁS 1990, P. 1FF], [CHARTRAND & OELLERMANN 1993, P. 6], [KUSIAK 1999, P. 36FF], [MELNIKOV ET AL. 1994, P. 275FF]

7.1.11 Isolated node

Definition

An isolated node does not possess any edges to or from other nodes of the considered structure.



Significance in static networks

Adaptations to the isolated node do not affect other nodes of the structure. User-defined adaptations to nodes of the structure do not impact the isolated node.

Significance in dynamic networks

An isolated node is not integrated in a flow of activities and can be processed independently from all nodes of the structure considered.

Challenge of structure improvement

If structural constellations are considered in a multi-domain environment, nodes can be isolated in specific views while being integrated into system structures in other ones.

Potential

In many cases, an isolated node can be excluded from further considerations of the structure. Isolated nodes often appear in structures that emerge from the combination of system views that have been separately generated so far. This can, for example, indicate a lack of interface definitions; however, especially in a multi-domain environment, interpretation of isolated nodes depends on the specific meaning of the nodes and edges considered.

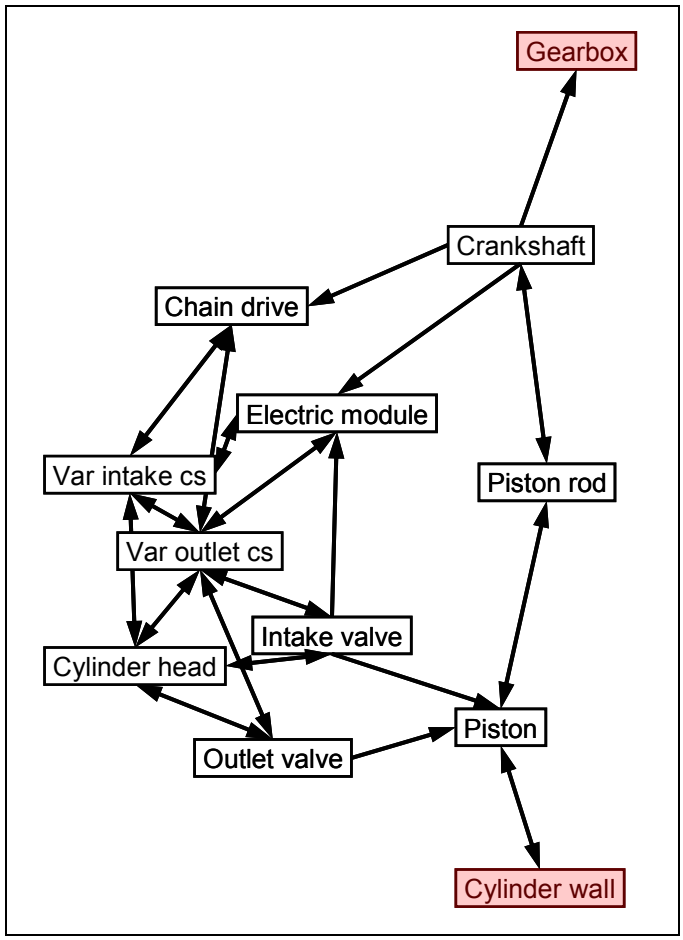
Literature

[BOLLOBÁS 1990, P. 3], [MELNIKOV ET AL. 1994, P. 22],
[CHARTRAND & OELLERMANN 1993, P. 6]

7.1.12 Leaf

Definition

The criterion leaf indicates a node which is only connected to one other node in a structure. Additionally, leaves can be differentiated by active, passive, and bi-directional leaves.



Significance in static networks

An active leaf only directly affects one node. A passive leaf is only affected by one single node. Adaptations to the leaf and the node directly linked to it can often be considered simultaneously.

Significance in dynamic networks

A leaf and its connected node represent a detailed modeling of a superior activity and typically do not contain information that is relevant for structure considerations.

Challenge of structure improvement

Nodes can represent leaves in a specific subset but possess dependencies in the entire structure. That means that the correct selection of the system part considered is of major importance for obtaining significant analysis results.

Potential

Leaves can often be combined with the node directly connected to them for simplification of structural considerations.

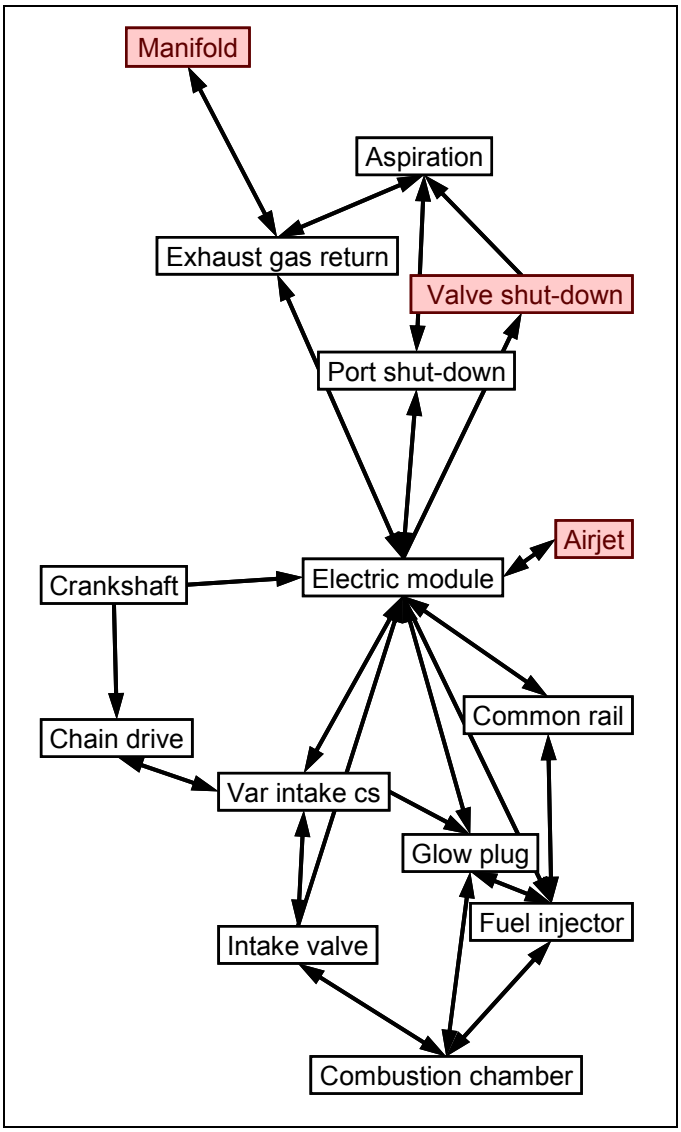
Literature

[ANDRÀSFAI 1991, P. 4FF], [CHARTRAND & OELLERMANN 1993, P. 70FF]

7.1.13 Transit node

Definition

A transit node is characterized by exactly one incoming and one outgoing dependency. A specific case is the connection of a node by only one bi-directional dependency.



Significance in static and dynamic networks

A transit node only affects one node directly and becomes only affected by one node directly as well. A transit node and one of the directly linked nodes can often be considered as an integral unit without disregarding relevant structural information. In this case the transit node and a linked node represent a detailed modeling of a superior node.

Challenge of structure improvement

Nodes may represent transit nodes in a specific subset but possess more dependencies in the entire structure. That means that the correct selection of the system part considered is of major importance for obtaining significant analysis results.

Potential

Transit nodes can often be combined with one of the two directly linked nodes and disregarded for other aspects of structure modeling. That allows further analyses to be simplified, because the quantity of system elements can be decreased without exclusion or falsification of structural constellations.

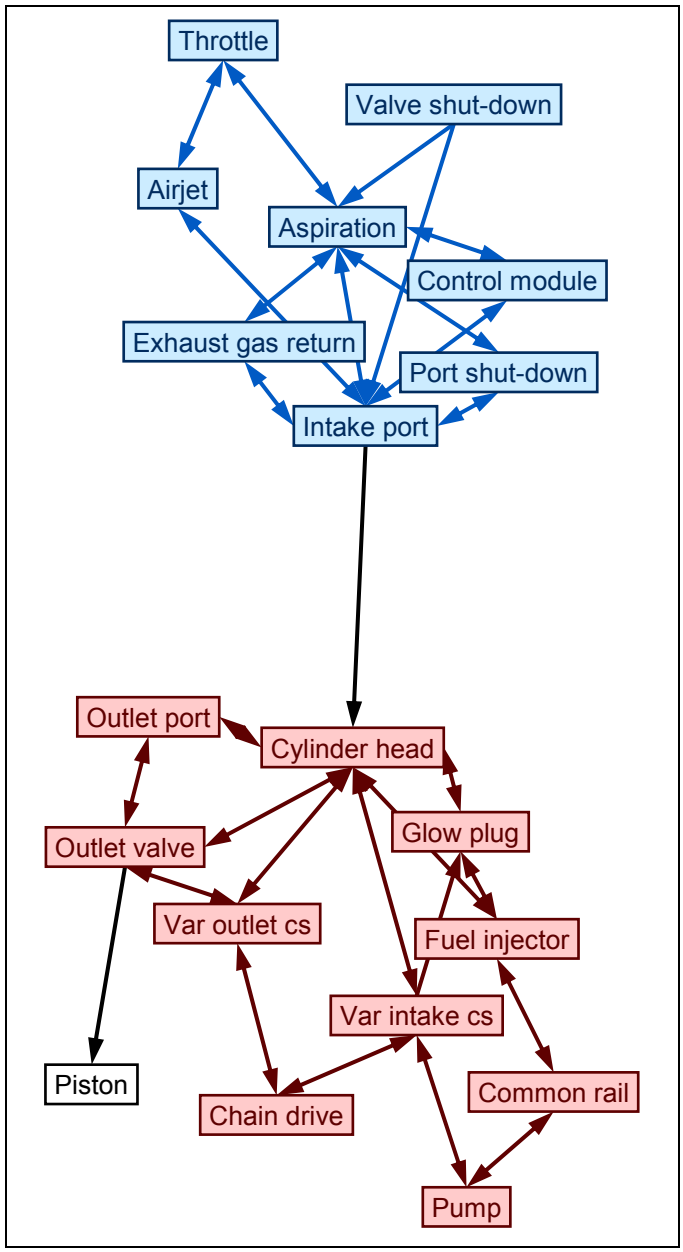
Literature

[MELNIKOV ET AL. 1994, P. 22]

7.2 Basic analysis criteria characterizing subsets

In the following, twelve basic analysis criteria are listed in alphabetical order. These criteria allow subsets (comprising nodes and edges) to be characterized in a dependency network. Relevant network elements are highlighted in color in the graphical examples.

7.2.1 Bi-connected component



Definition

A bi-connected component describes the subset of a graph, where all nodes are linked to each other (not imperatively by direct dependencies) and the removal of one user-defined dependency does not break up the general connectivity of the subset.

Significance in static and dynamic networks

None of the edges in a bi-connected component represents an exclusive connection to any included node.

Challenge of structure improvement

Bi-connected components occur mostly in highly connected networks and their identification generally requires algorithmic support. The existence of bi-connected components often accompanies the existence of several other structural analysis criteria, e.g., feedback loops imperatively exist in bi-connected components of non-directed graphs.

Potential

The existence of bi-connected components is important for system reliability, as the removal of one edge does not completely impede the interaction between the nodes of the bi-connected component. A bi-connected component can be opposite a modular design, if the bi-connected component stretches across the planned parting line of two modules.

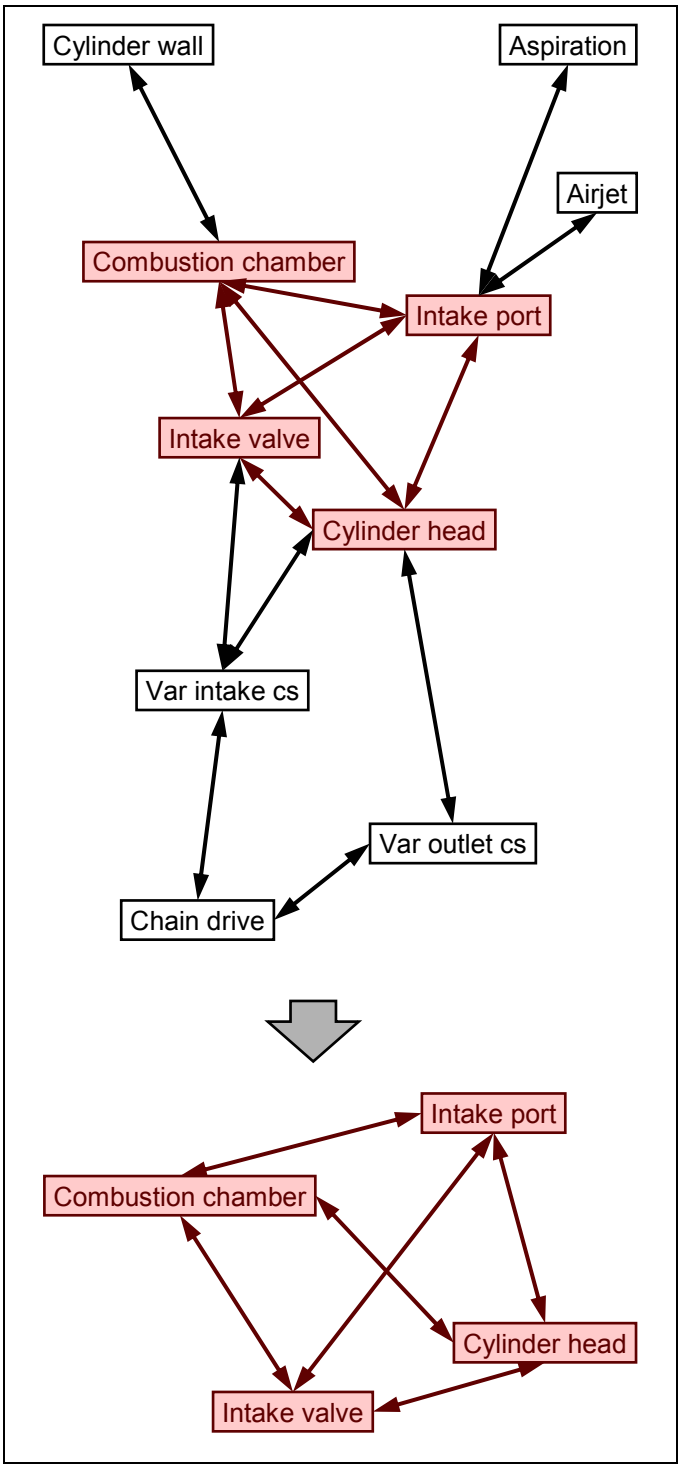
Literature

[MELNIKOV ET AL. 1994, P. 138FF]

7.2.2 Cluster (completely cross-linked)

Definition

A completely cross-linked cluster represents a structural subset that possesses the maximum amount of internal dependencies between the nodes of the cluster and only a limited number of external dependencies to the rest of the structure considered.



Significance in static and dynamic networks

All nodes of a completely cross-linked cluster directly impact each other.

Challenge of structure improvement

If nodes of a structure are highly connected, clusters may overlap in nodes and edges. In this case the identification of clusters asks for algorithmic support, as e.g., the visual alignment of one cluster in a matrix depiction can hide another one. The existence of a cluster accompanies further structural criteria, e.g., the existence of feedback loops.

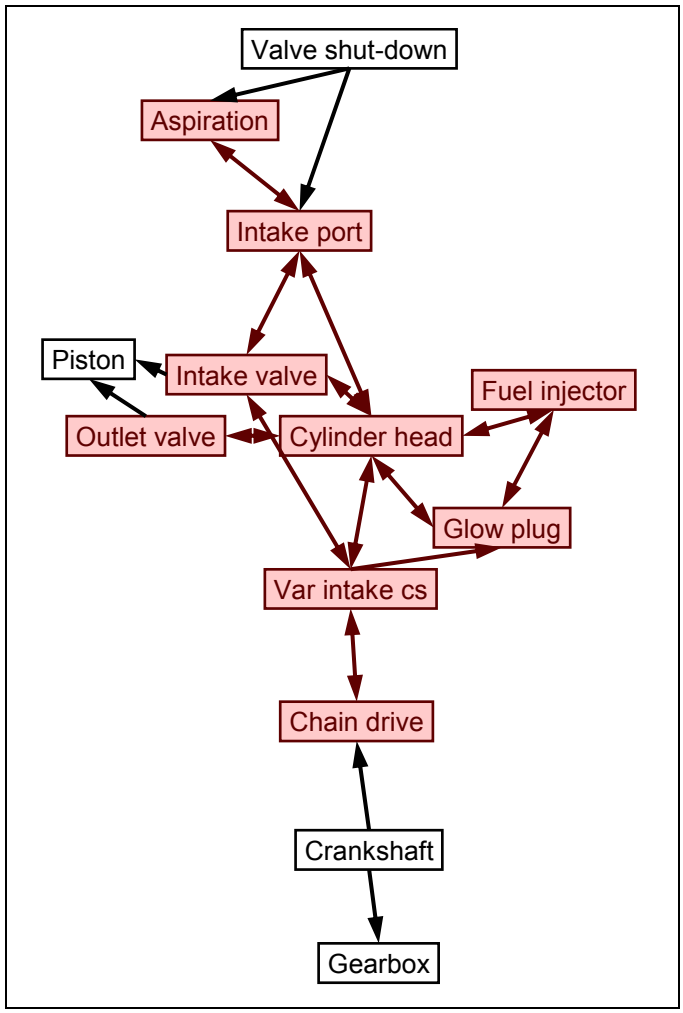
Potential

Completely cross-linked clusters represent subsets that are suitable for the definition of modules. Depending on additional system views (e.g., functional dependencies) the scope of a defined cluster can be optimized (integration or extraction of specific nodes).

Literature

[ANDRÀSFAI 1991, P. 12FF], [HARTIGAN 1975, P. 12FF]

7.2.3 Cluster (based on a strongly connected part)



Definition

A cluster that is based on a strongly connected part represents a structural subset that possesses a high amount of internal dependencies compared to its number of external dependencies within the rest of the structure. As the cluster is formed by nodes that fulfill the criterion of a strongly connected part, all nodes in such a cluster are connected at least indirectly by dependency paths.

Significance in static and dynamic networks

All nodes of a cluster that is based on a strongly connected part impact each other at least indirectly.

Challenge of structure improvement

The identification of clusters often asks for algorithmic support, as for example, the manual alignment of a cluster in large matrices can be difficult. The existence of a cluster is often accompanied by further structural criteria, e.g., every strongly connected part comprises at least one feedback loop.

Potential

Clusters that are based on strongly connected parts represent subsets suitable for the definition of modules. Depending on additional system views (e.g., functional dependencies), the scope of a defined cluster can be optimized (the integration or extraction of specific nodes).

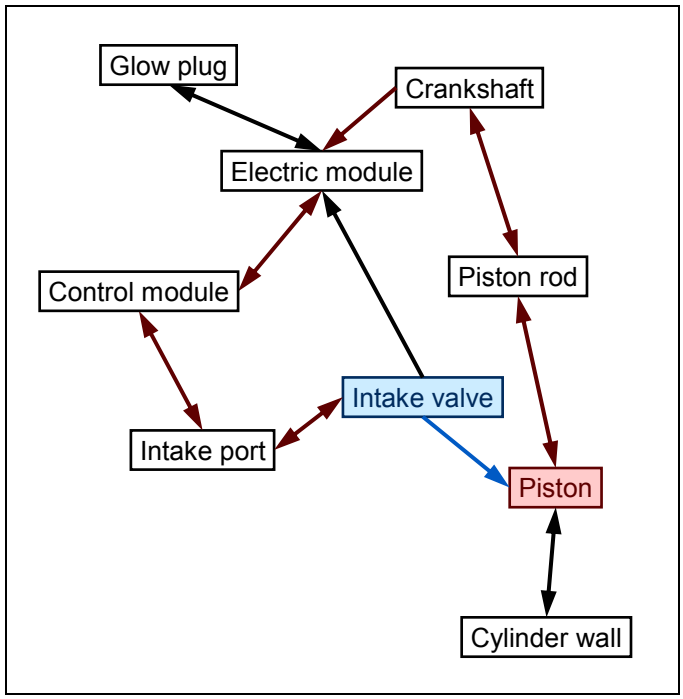
Literature

[HARTIGAN 1975, P. 12FF], [CHARTRAND & OELLERMANN 1993, P. 319FF]

7.2.4 Distance (between nodes)

Definition

The distance between two nodes describes the path that contains the minimum number of dependencies between them. The distance criterion takes into account the direction of the dependencies.



Significance in static networks

Short distances mean direct impact between nodes. The longer a distance becomes, the more likely that adaptations to the initial node will not affect the second one due to moderation effects. However, in the case of long distances between nodes, the possible resulting effects may be unknown to users.

Significance in dynamic networks

Depending on the specific modeling content, short distances between two nodes can indicate that activities have to follow each other closely in time.

Challenge of structure improvement

Individual distances can be determined visually by use of graph representations or by algorithmic support. General analysis of distances between nodes of a structure results in major computational efforts and requires effective software support.

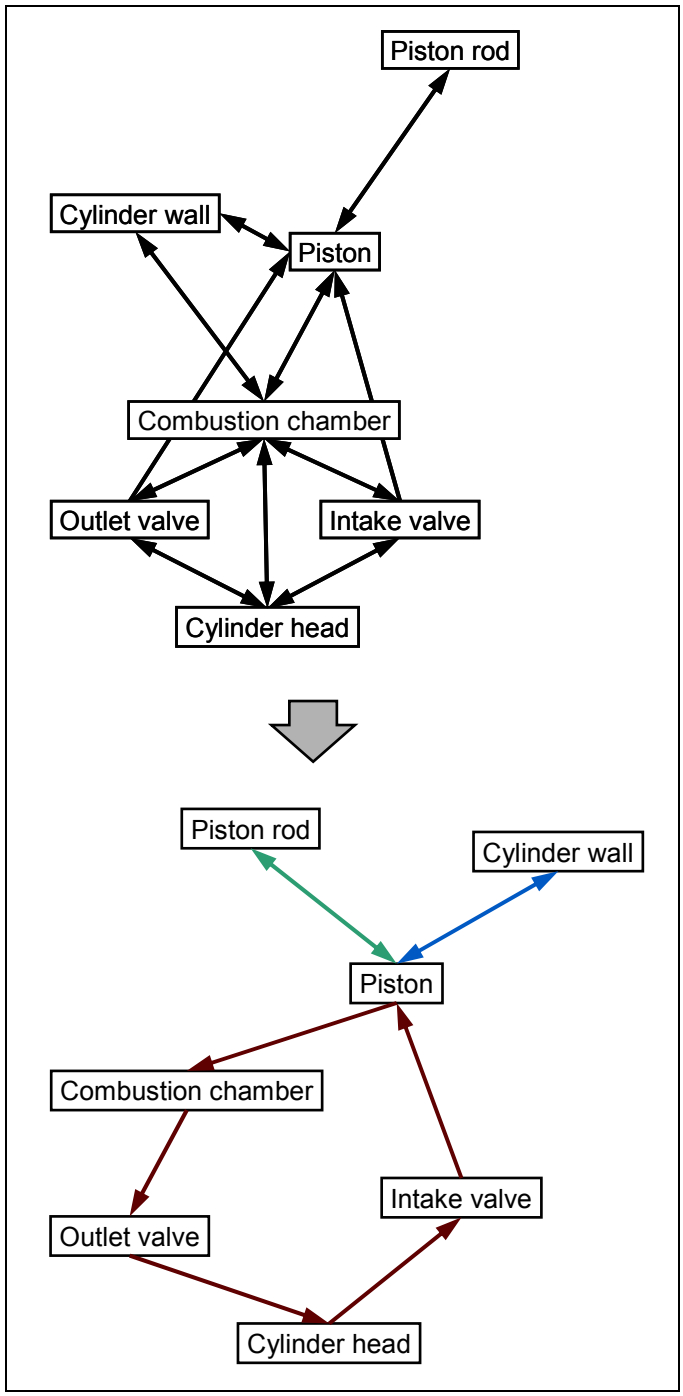
Potential

Distances between nodes have to correspond to, for example, functional or coordination requirements in a system. Analyses of distances allow for the identification of specific system nodes, whose integration to the entire structure has to be adapted.

Literature

[CHARTRAND & OELLERMANN 1993, P. 99FF], [FREEMAN 1979]

7.2.5 Feedback loop



Definition

A feedback loop consists of two or more nodes that are interlocked sequentially by edges and therefore reciprocally influence each other.

Significance in static networks

A feedback loop can provide self-energizing or self-impeding effects once an element of the loop is adapted.

Significance in dynamic networks

A feedback loop means the necessity of iteratively processing nodes, as the output of one node that is scheduled later in the structural alignment (and obtains input from preceding nodes) is required as input for a preceding node.

Challenge of structure improvement

Algorithmic support is generally required for feedback loop analysis. A large number of feedback loops can exist in highly interconnected structures. Often feedback loops overlap each other in nodes and edges.

Potential

The adaptation of nodes and edges that participate in several feedback loops simultaneously can significantly influence their overall quantity. Generally, with feedback loop reduction, structures that are easier to manage are achieved.

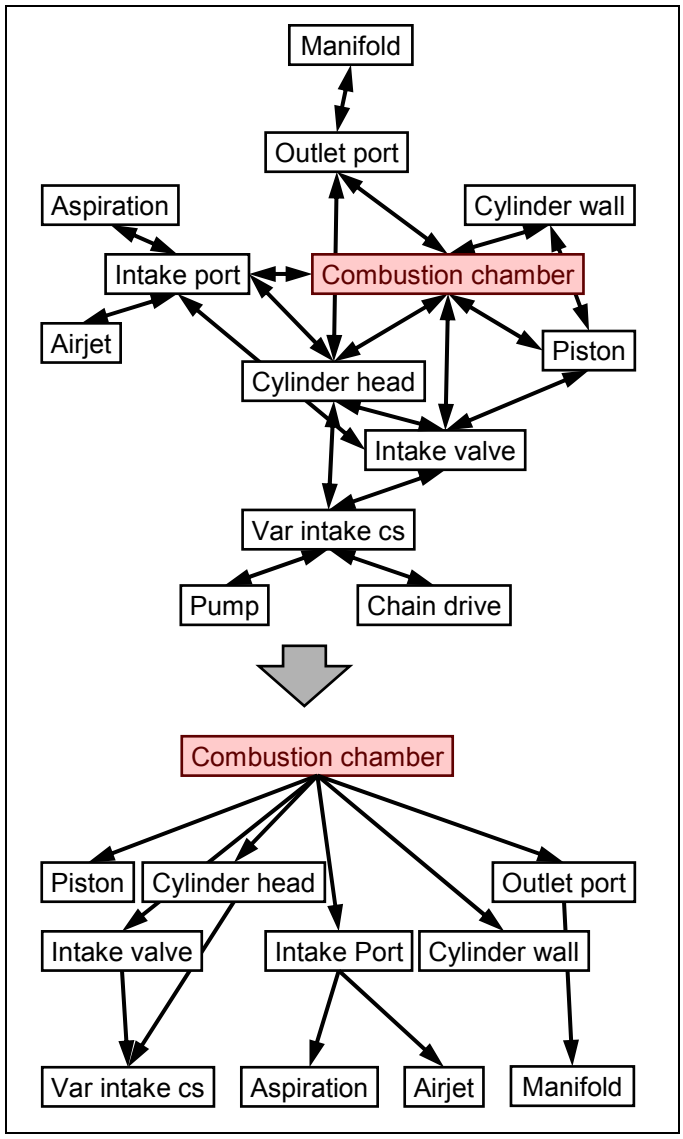
Literature

[FOULDS 1992, P. 38FF], [KUSIAK 1999, P. 36FF], [MELNIKOV ET AL. 1994, P. 18FF]

7.2.6 Hierarchy

Definition

A hierarchy subset is characterized by a dependency chain starting from a top node and expanding over several levels of a multitude of nodes. A hierarchy subset does not contain dependencies reaching from subordinated elements back to superior ones.



Significance in static networks

The element that is on the highest hierarchical level can induce a cascade of change impacts on all subsequent components in case of its adaptation.

Significance in dynamic networks

The element on the top level of a hierarchy induces many subsequent activities.

Challenge of structure improvement

Algorithmic support for computation and visualization of hierarchy subsets is generally required. A high quantity of such hierarchical subsets can occur in a structure; often hierarchies overlap each other in practical applications. Changes to single edges and nodes can influence the extent of several hierarchy subsets within a structure.

Potential

The adaptation of specific edges often significantly influences the extent of hierarchies in a structure. Nodes of hierarchical subsets can be manipulated effectively by adapting the top element. In contrast, undesired changes to top elements can result in extensive system chances.

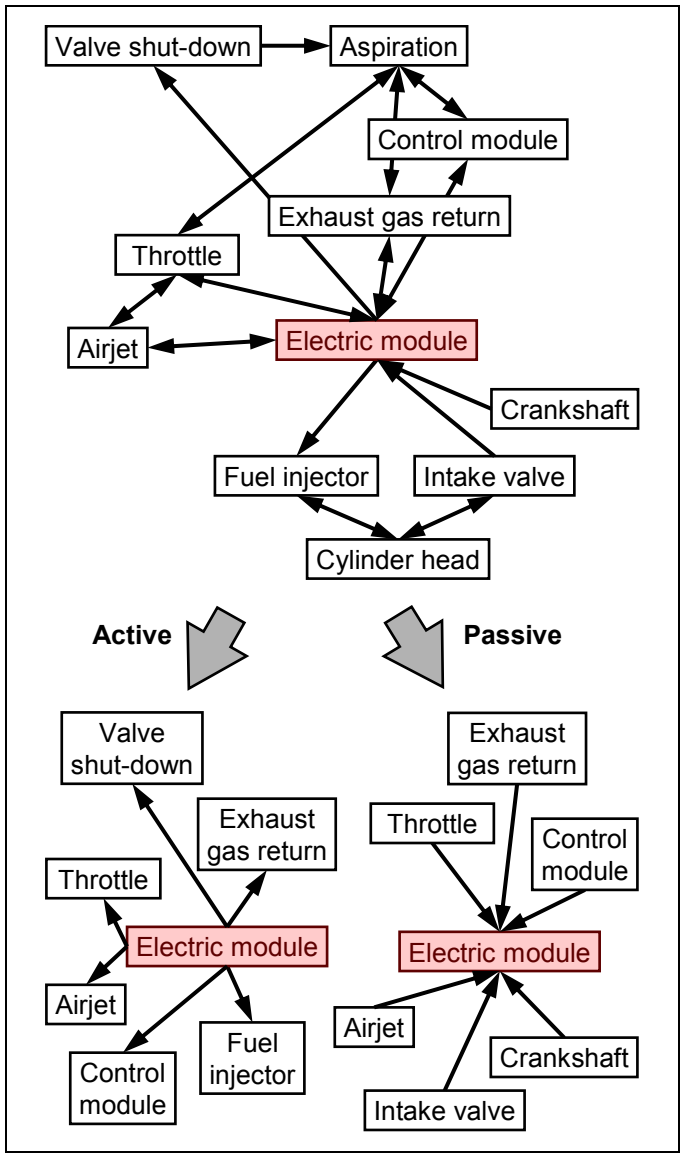
Literature

[FOULDS 1992, P. 27FF], [MELNIKOV ET AL. 1994, P. 51FF]

7.2.7 Locality

Definition

The locality depicts the surrounding of a specific node, i.e., all nodes that are linked directly to the node in question. The active (passive) locality only comprises the outgoing (incoming) dependencies of a specific node.



Significance in static networks

The active locality of a node represents all nodes that can be impacted directly by adaptations to the node in question. The passive locality represents all nodes that can provide direct impact to the node in question.

Significance in dynamic networks

The active locality of a node represents its succeeding nodes in the structure. In contrast, the passive locality represents nodes directly preceding it in the structure.

Challenge of structure improvement

The locality of a node can be depicted, for example, in graph and matrix visualization without extensive effort. Different localities can exist for the same node depending on the structural subset considered and the level of detail.

Potential

The locality criterion is helpful for evaluating the direct structural embedding of specific nodes to the structure and helps identify their behavior in a system. Observations of the locality are not limited to the directly linked nodes of the first level, but, for example, can be extended to nodes that can be reached by two dependencies.

Literature

[CHARTRAND & OELLERMANN 1993, P. 100FF]

7.2.8 Path

Definition

A path represents a direct or indirect connection of two nodes by edges. Specific paths are the shortest and longest paths as well as paths fulfilling selected constraints (e.g., passing by a defined third node).

Significance in static networks

A path represents a possible impact chain between two nodes.

Significance in dynamic networks

A path represents a required sequence of activities.

Challenge of structure improvement

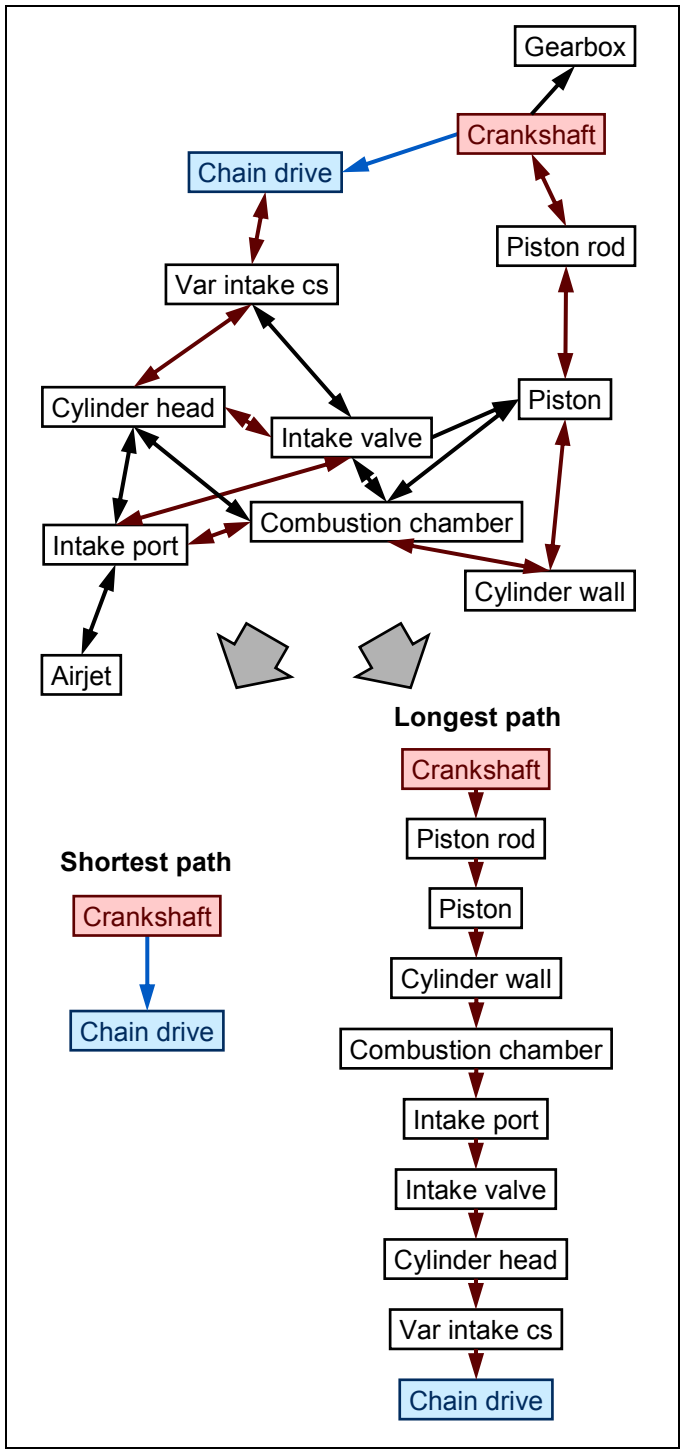
The identification of specific paths in comprehensive networks generally requires algorithmic support. Often constraints on the desired result are mandatory for confining resulting paths to a manageable amount.

Potential

If far-reaching indirect dependencies between nodes are not systematically investigated, they often remain unconsidered.

Literature

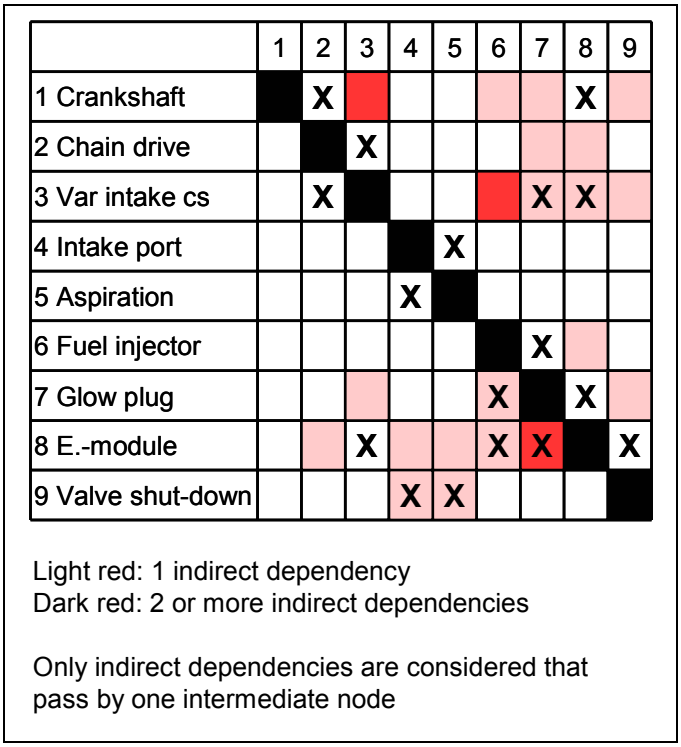
[ANDRÀSFAI 1991, P. 30FF],
[MELNIKOV ET AL. 1990, P. 286FF],
[THULASIRAMAN & SWAMY 1992, P. 314FF]



7.2.9 Quantity of indirect dependencies

Definition

The quantity of indirect dependencies describes the number of indirect paths between two specific nodes. The extent of the indirect dependencies considered (length of paths) has to be defined.



Significance in static and dynamic networks

Even if two nodes are not connected directly by an edge, they can significantly impact each other by several indirect dependency paths.

Challenge of structure improvement

In structures that possess a high connectivity, the computation of the quantity of indirect dependencies particularly requires algorithmic support. Depending on the use case and the system aspects considered, a high or low quantity of indirect dependencies is desired. The identified quantity has to be rated after examining other existing analysis criteria (e.g., clusters or hierarchies)

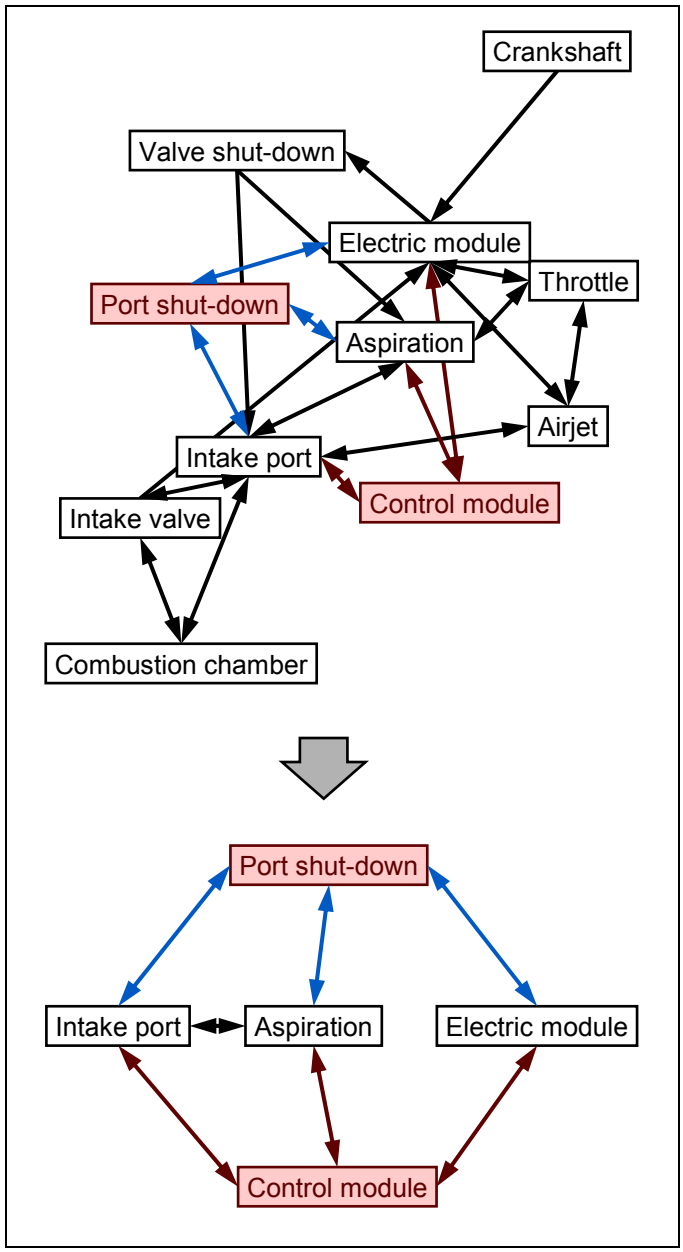
Potential

The quantity of indirect dependencies allows plausibility checks for the acquired dependencies of a system. Noticeably high or low values can suggest a closer examination of specific network parts. This criterion also allows nodes that are highly dependent on each other, but often do not possess a direct dependency, to be identified.

Literature

[EICHINGER ET AL. 2006]

7.2.10 Similarity



Definition

The similarity of two nodes describes the amount of identical connections both nodes possess to surrounding nodes in the structure. The criterion can be further classified into active and passive similarity comparing only outgoing or incoming edges. Additionally, similarity can refer to existing as well as non-existing edges.

Significance in static networks

A high active (passive) similarity means that both nodes provide (obtain) impact to (from) several identical nodes in the structure.

Significance in dynamic networks

A high active (passive) similarity of nodes means that they possess several identical successors (predecessors) or identical receivers of the nodes' output (providers of nodes' input).

Challenge of structure improvement

In structures that possess a high connectivity, the computation of the similarity generally requires algorithmic support. Depending on the use case, a high or low similarity is desired.

Potential

Nodes' similarity can provide useful insight into a system structure and the behavior of its nodes, especially if further structure criteria are difficult to identify.

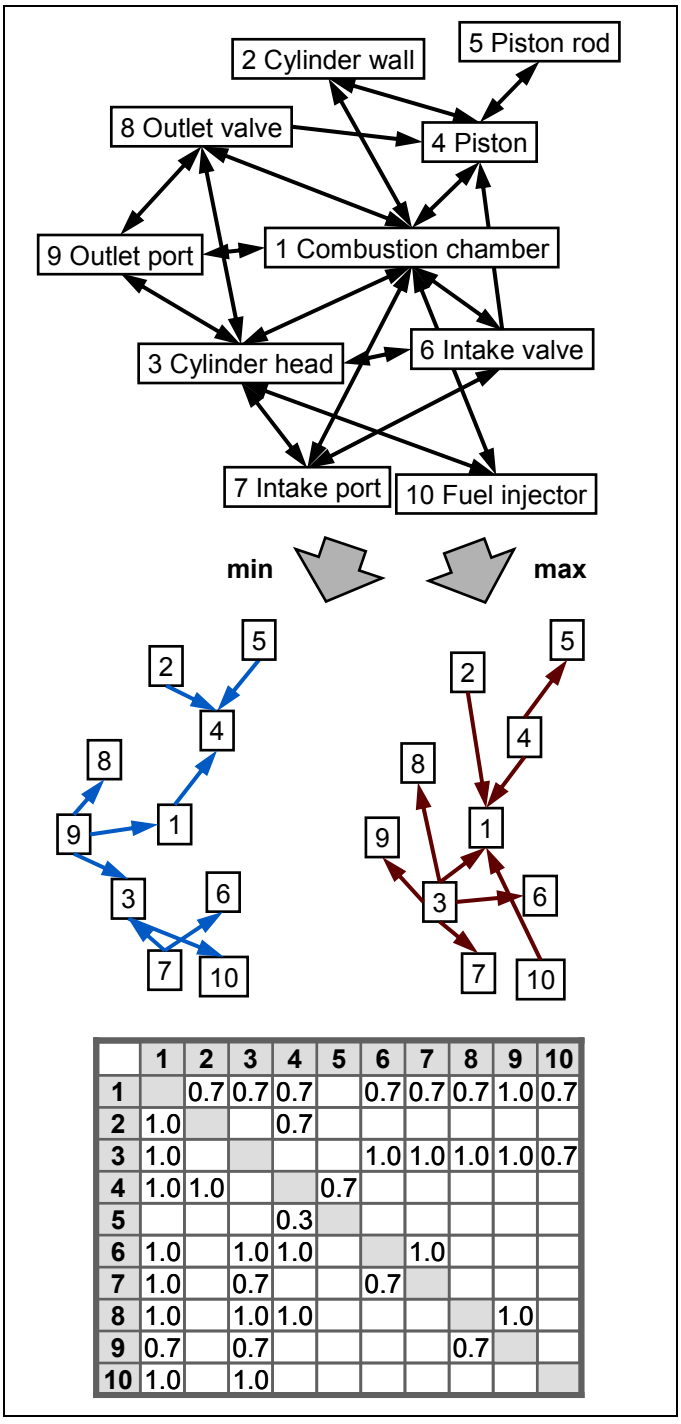
Literature

[DANILOVIC & BROWNING 2004], [MCCORMICK ET AL. 1972]

7.2.11 Spanning tree

Definition

A spanning tree represents a subset of a structure and contains all nodes but only a minimum of the existing edges required for retaining the coherence of the entire structure. Specifications of the criterion are the minimum and the maximum spanning. The minimum (maximum) spanning tree selects edges of the structure that result in a minimum (maximum) sum of edge weights.



	1	2	3	4	5	6	7	8	9	10
1		0.7	0.7	0.7		0.7	0.7	0.7	1.0	0.7
2	1.0			0.7						
3	1.0					1.0	1.0	1.0	1.0	0.7
4	1.0	1.0			0.7					
5				0.3						
6	1.0		1.0	1.0			1.0			
7	1.0		0.7			0.7				
8	1.0		1.0	1.0					1.0	
9	0.7		0.7					0.7		
10	1.0		1.0							

Significance in static networks

A spanning tree describes a core structure of the system, which is at minimum required without dissolving its integrity.

Significance in dynamic networks

A minimum spanning tree can represent the fastest way of propagation between nodes of a system.

Challenge of structure improvement

In structures that possess a high connectivity in particular, the computation of spanning trees requires algorithmic support. Depending on the use case, minimum or maximum spanning trees (or others based on further specific constraints) are desired.

Potential

Depending on the meaning of edges and nodes, a spanning tree can represent a core structure of a system.

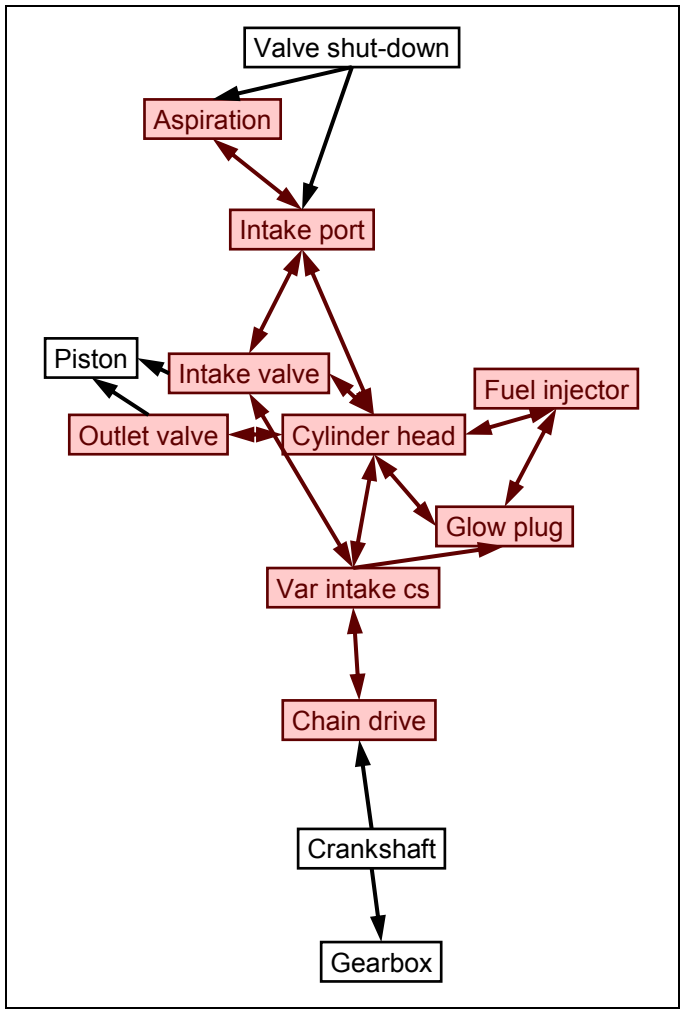
Literature

[FOULDS 1992, P. 37F], [MELNIKOV ET AL. 1994, P. 58FF]

7.2.12 Strongly connected part/component

Definition

A strongly connected part characterizes a subset, where every node can be reached by all other nodes at least by one dependency path. Strongly connected parts represent an enhancement of the definition of completely cross-linked clusters, because all nodes of the subset do not have to be directly linked to each other but are indirectly linked.



Significance in static networks

A strongly connected part describes a collection of components that can all mutually affect the others if they are adapted.

Significance in dynamic networks

All activities that constitute part of a strongly connected part require (partly indirect) input from each other and therefore have to be executed simultaneously and must be permanently coordinated.

Challenge of structure improvement

In structures that possess a high connectivity, in particular, the computation of strongly connected parts requires algorithmic support.

Potential

Strongly connected parts are suitable for defining the extent of modules or building blocks.

Literature

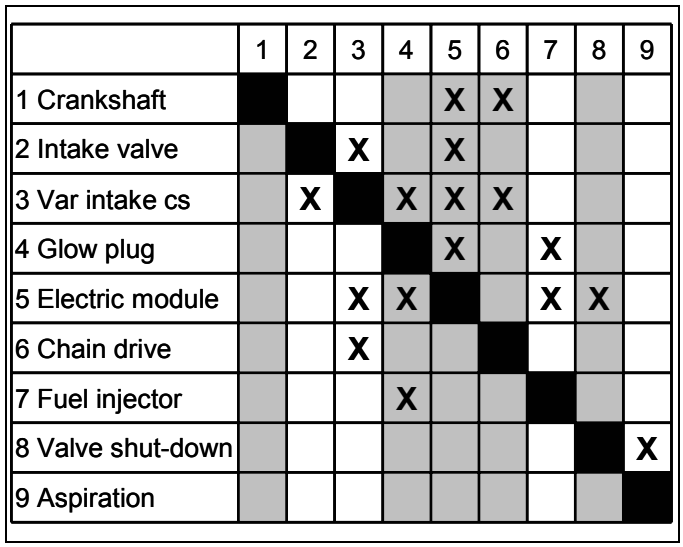
[CHARTRAND & OELLERMANN 1993, P. 319FF], [ANDRÀSFAI 1991, P. 19FF], [THULASIRAMAN & SWAMY 1992, P. 354FF]

7.3 Basic analysis criteria characterizing graphs

In the following, six basic analysis criteria are listed in alphabetical order. These criteria allow entire graph structures to be characterized. The criteria make use of further criteria that specify nodes, edges, and subsets and are described in the preceding Sections 7.1 and 7.2. Relevant network elements are highlighted in color in the graphical examples.

7.3.1 Banding

Definition



	1	2	3	4	5	6	7	8	9
1 Crankshaft	■				X	X			
2 Intake valve		■	X		X				
3 Var intake cs		X	■	X	X	X			
4 Glow plug				■	X		X		
5 Electric module			X	X	■		X	X	
6 Chain drive			X			■			
7 Fuel injector				X			■		
8 Valve shut-down								■	X
9 Aspiration									■

Banding represents an enhancement of the partitioning analysis and is meant to indicate mutually independent groups of nodes in a structure by the realignment of a matrix depiction. Nodes that can be processed independently from each other and that are arranged side by side constitute part of the same band. Bands are typically highlighted by light and dark colored matrix columns. The Banding approach does not consider backward edges (edges below the matrix diagonal) and only analyzes possible dependencies between adjoined bands.

Significance in static networks

Banding can indicate the independence between some elements' groups in case feedback loops do not exist or can be excluded from consideration.

Significance in dynamic networks

The bands of a matrix can allow the possibility for enhancement of parallel process design to be identified in case feedback loops do not exist or can be excluded from consideration.

Challenge of structure improvement

The objective is to obtain as few bands in a matrix as possible. Banding refers to the different hierarchical levels identified by partitioning a matrix description; however, the occurrence of feedback loops is ignored. As activity networks are often affected by iterations in practical use cases, the useful applicability of banding in complex systems can be questioned, as the analysis is based on excluding important structural characteristics from considerations.

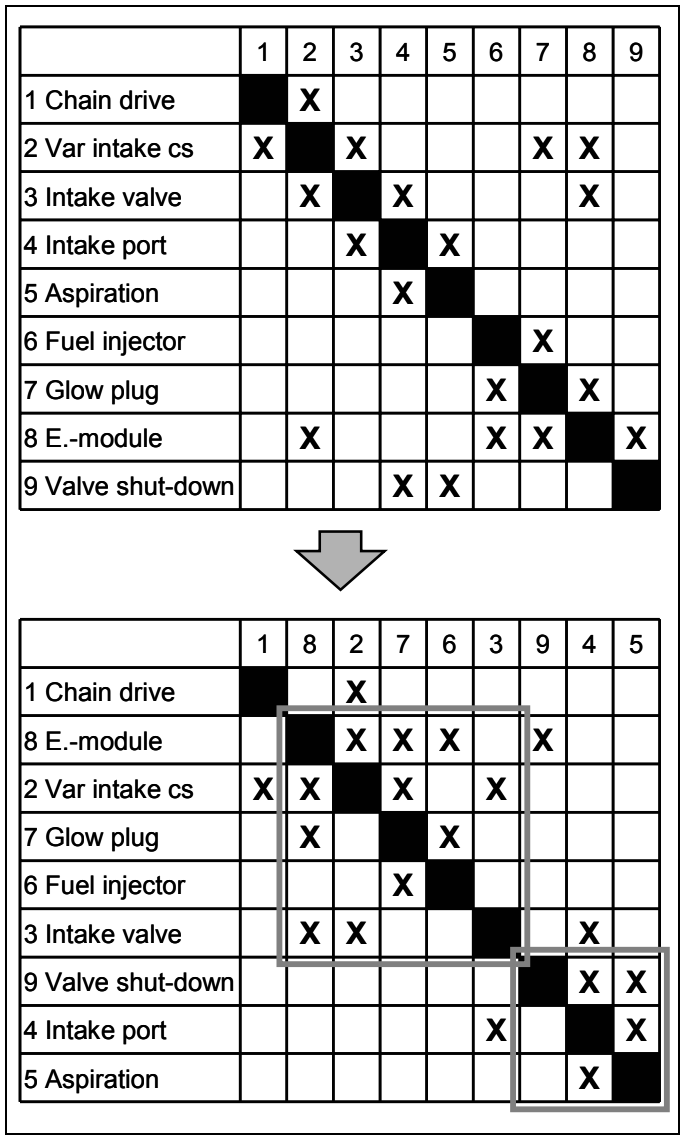
Potential

Banding can shorten an overall process time by identification of the activities that can be parallelized. Groups of nodes that are mutually independent can also be identified if feedback loops are not relevant for consideration.

Literature

[GROSE 1994], [YASSINE 2004]

7.3.2 Clustering



	1	2	3	4	5	6	7	8	9
1 Chain drive		X							
2 Var intake cs	X		X				X	X	
3 Intake valve		X		X				X	
4 Intake port			X		X				
5 Aspiration				X					
6 Fuel injector							X		
7 Glow plug						X		X	
8 E.-module		X				X	X		X
9 Valve shut-down				X	X				

	1	8	2	7	6	3	9	4	5
1 Chain drive			X						
8 E.-module			X	X	X		X		
2 Var intake cs	X	X		X		X			
7 Glow plug		X			X				
6 Fuel injector				X					
3 Intake valve		X	X					X	
9 Valve shut-down								X	X
4 Intake port						X			X
5 Aspiration								X	

Definition

The objective of the clustering analysis is to detect subsets that possess many internal dependencies between nodes of a cluster subset and as few dependencies as possible to or from the external nodes of the structure.

Significance in static networks

A cluster represents an appropriate basis for creating a module: an adaptation to one node of the cluster causes many impacts to other nodes in this subset, whereas the replacement of the whole cluster (module) requires only the consideration of a few dependencies to other nodes.

Significance in dynamic networks

A cluster generally represents nodes that can be executed simultaneously with intense coordination because of their mutual connection.

Challenge of structure improvement

In structures that possess many nodes and a high connectivity, in particular, clustering requires algorithmic support. Many different clustering strategies and algorithms are available and possess different suitability for specific use cases. If clusters overlap each other, matrix depictions can only be partially applied for representing resulting clusters.

Potential

Clustering can allow the identification of closely related groups of nodes and their mutual overlapping. This provides the basis for modularization strategies.

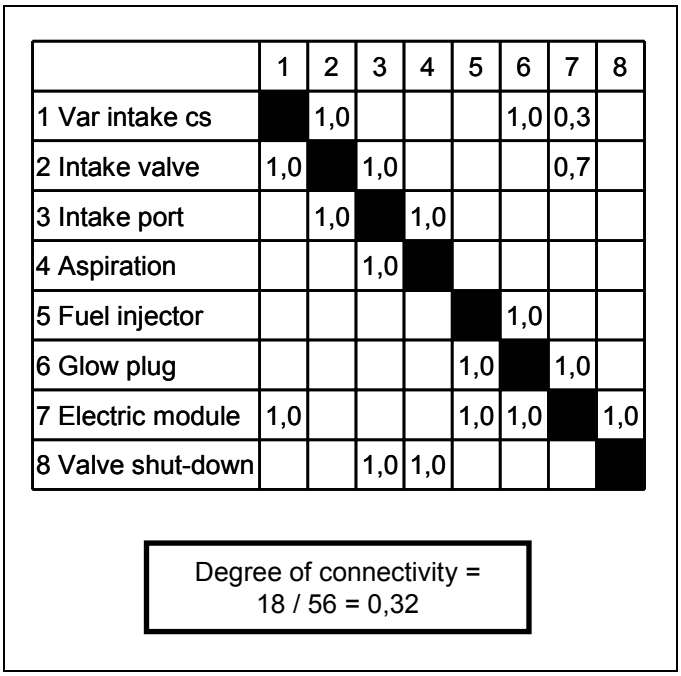
Literature

[BROWNING 2001], [KUSIAK 2000, P. 294FF], [PIMMLER & EPPINGER 1994], [SHARMAN & YASSINE 2004], [YU ET AL. 2003]

7.3.3 Degree of connectivity

Definition

The degree of connectivity represents the ratio of existing edges to the quantity of all possible edges in a structure.



	1	2	3	4	5	6	7	8
1 Var intake cs		1,0				1,0	0,3	
2 Intake valve	1,0		1,0				0,7	
3 Intake port		1,0		1,0				
4 Aspiration			1,0					
5 Fuel injector						1,0		
6 Glow plug					1,0		1,0	
7 Electric module	1,0				1,0	1,0		1,0
8 Valve shut-down			1,0	1,0				

Degree of connectivity = 18 / 56 = 0,32

Significance in static and dynamic networks

The degree of connectivity can indicate the accuracy of the level of detail of the model.

Challenge of structure improvement

The degree of connectivity is easy to determine for any user-defined structures but can only lead to significant structure interpretation when it is considered in combination with further structural criteria.

Potential

The degree of connectivity can provide plausibility checks and the selection of appropriate analysis criteria as well as the chosen level of detail. A compromise has to be found considering the right level of detail, the degree of connectivity, and a manageable quantity of system elements. As similar products typically possess similar structures, comparison of the degree of connectivity can be an initial indication of the quality of the available data after information acquisition. A high degree of connectivity can make the application of specific analysis impossible. Thus, the quantity of feedback loops can increase significantly when the connectivity of elements in a structure becomes higher.

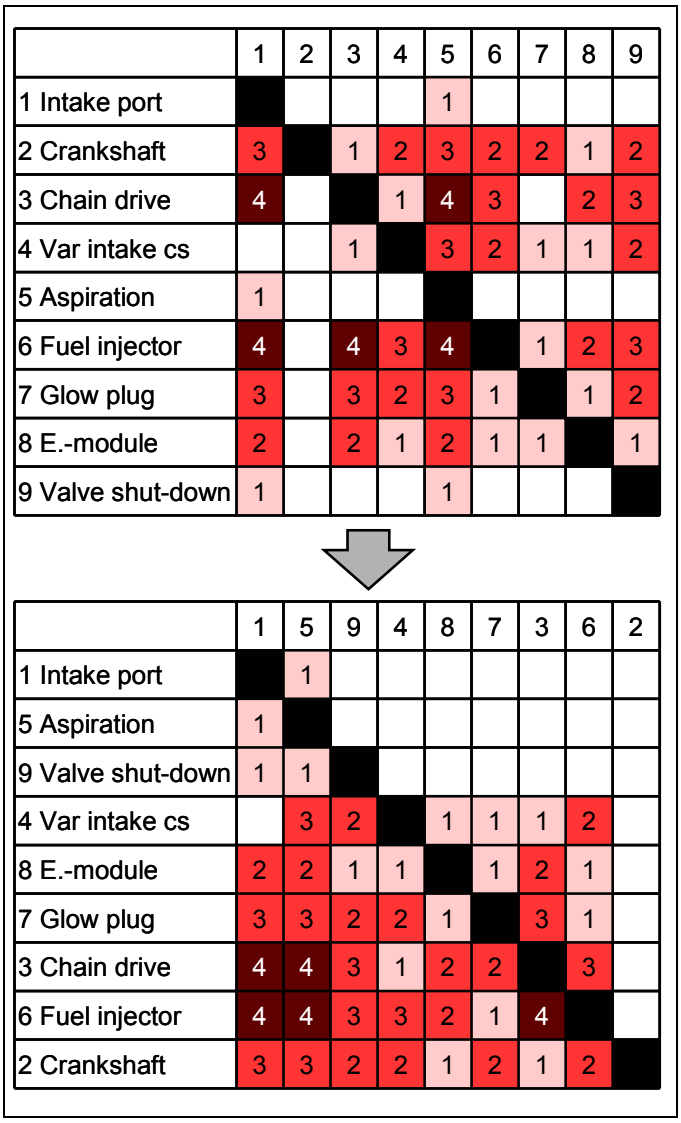
Literature

[EICHINGER ET AL. 2006], [PUHL 1999, P. 29F]

7.3.4 Distance matrix

Definition

The distance matrix applies the distance criterion and depicts the resulting values for all possible node pairings in the cells of a DSM. The rows and columns of the matrix can then be realigned in order to indicate blocks of similar distance values.



	1	2	3	4	5	6	7	8	9
1 Intake port					1				
2 Crankshaft	3		1	2	3	2	2	1	2
3 Chain drive	4			1	4	3		2	3
4 Var intake cs			1		3	2	1	1	2
5 Aspiration	1								
6 Fuel injector	4		4	3	4		1	2	3
7 Glow plug	3		3	2	3	1		1	2
8 E.-module	2		2	1	2	1	1		1
9 Valve shut-down	1				1				

	1	5	9	4	8	7	3	6	2
1 Intake port		1							
5 Aspiration	1								
9 Valve shut-down	1	1							
4 Var intake cs		3	2		1	1	1	2	
8 E.-module	2	2	1	1		1	2	1	
7 Glow plug	3	3	2	2	1		3	1	
3 Chain drive	4	4	3	1	2	2		3	
6 Fuel injector	4	4	3	3	2	1	4		
2 Crankshaft	3	3	2	2	1	2	1	2	

Significance in static networks

The distance matrix allows node groups to be identified that are closely related or, in contrast, only connected by far-reaching, indirect dependency chains.

Significance in dynamic networks

The distance matrix can indicate node groups according to their time-based order.

Challenge of structure improvement

When considering larger matrices, software support is recommended for computing distances and providing appropriate matrix realignment by application of cluster algorithms.

Potential

The distance matrix focuses on relevant node groups in order to evaluate any possible change impact to other system nodes or to determine the required input or output of related activities.

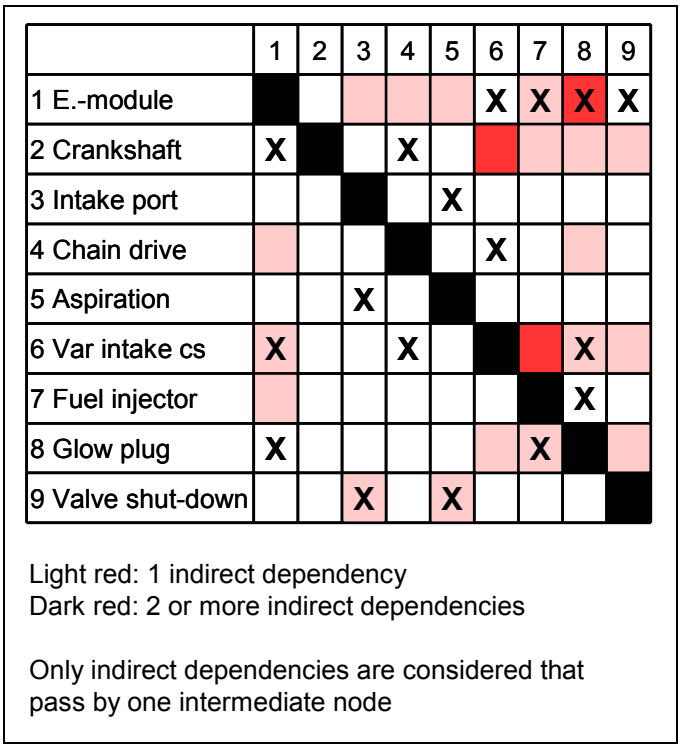
Literature

[EICHINGER ET AL. 2006]

7.3.5 Matrix of indirect dependencies

Definition

The matrix of indirect dependencies depicts the quantity of indirect paths that exist between each node pairing of a structure. The extent of indirect dependencies (length of paths) considered has to be defined.



Significance in static networks

The existence of a direct dependency often accompanies a high quantity of indirect dependencies between both nodes. This can be used to point out node pairings for closer consideration in the information acquisition process.

Challenge of structure improvement

The matrix of indirect dependencies is easy to determine for any user-defined structures but can only lead to significant structure interpretation when it is considered in combination with further structural criteria.

Potential

The application of the matrix of indirect dependencies provides an efficient means for plausibility checks during information acquisition. It can also help users acquire a general system understanding, as it is often difficult for developers to identify indirect dependencies in existing structures.

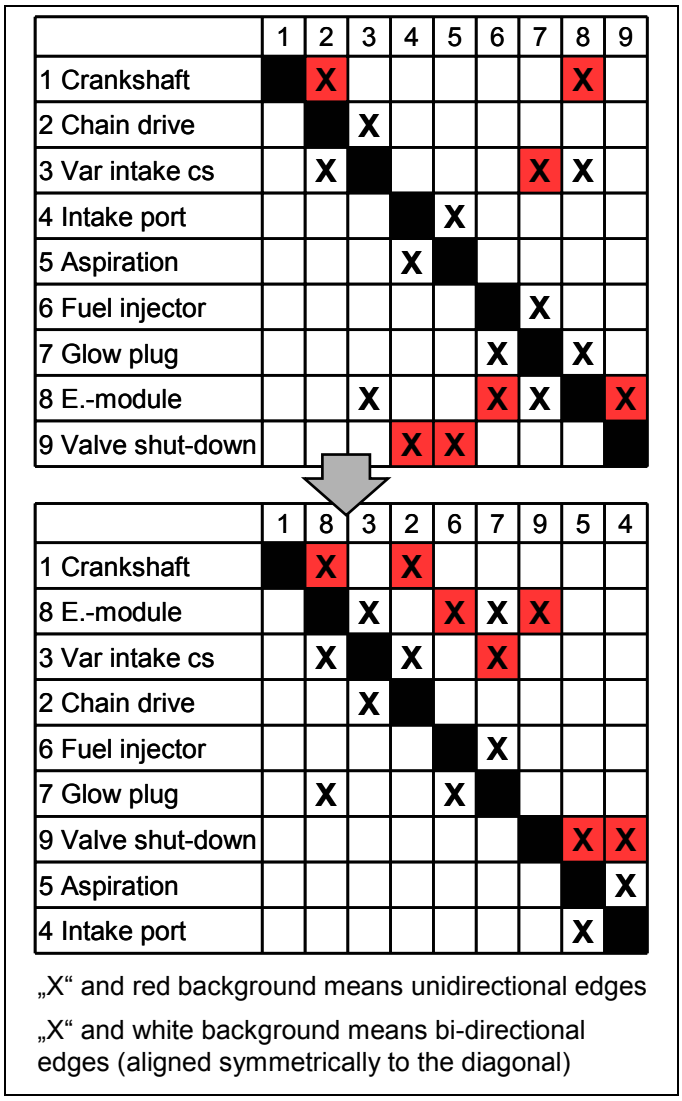
Literature

[EICHINGER ET AL. 2006]

7.3.6 Partitioning (triangularization, sequencing)

Definition

Partitioning describes the reordering of the rows and columns of a DSM with the objective of arranging all existing dependencies at one side or at least as close as possible to the diagonal. Other expressions for this approach are sequencing or triangularization.



Significance in static networks

Partitioning can indicate the existence of feedback loops. Groups of nodes can also be identified that are suitable for modular design.

Significance in dynamic networks

Partitioning can identify appropriate node sequences. A matrix alignment with all dependencies at one side of the diagonal means a process without any iteration steps.

Challenge of structure improvement

Often complex structures possess feedback loops that do not allow an alignment of edges at one side of the matrix diagonal. Partitioning then tries to align a minimum of edges below and all edges as close as possible to the diagonal. If loops overlap each other, common algorithms do not provide optimal alignment.

Potential

Partitioning can provide knowledge about the general existence of feedback loops (but not about specific loops) and can determine the strongly connected parts implied in a structure. Optimized orders for specifying components in static networks and optimized sequential arrangement for tasks in dynamic networks can be identified.

Literature

[Browning 2001], [Gebala & Eppinger 1991], [Kusiak 1999, p. 36ff], [Sharman & Yassine 2004], [Warfield 1973]

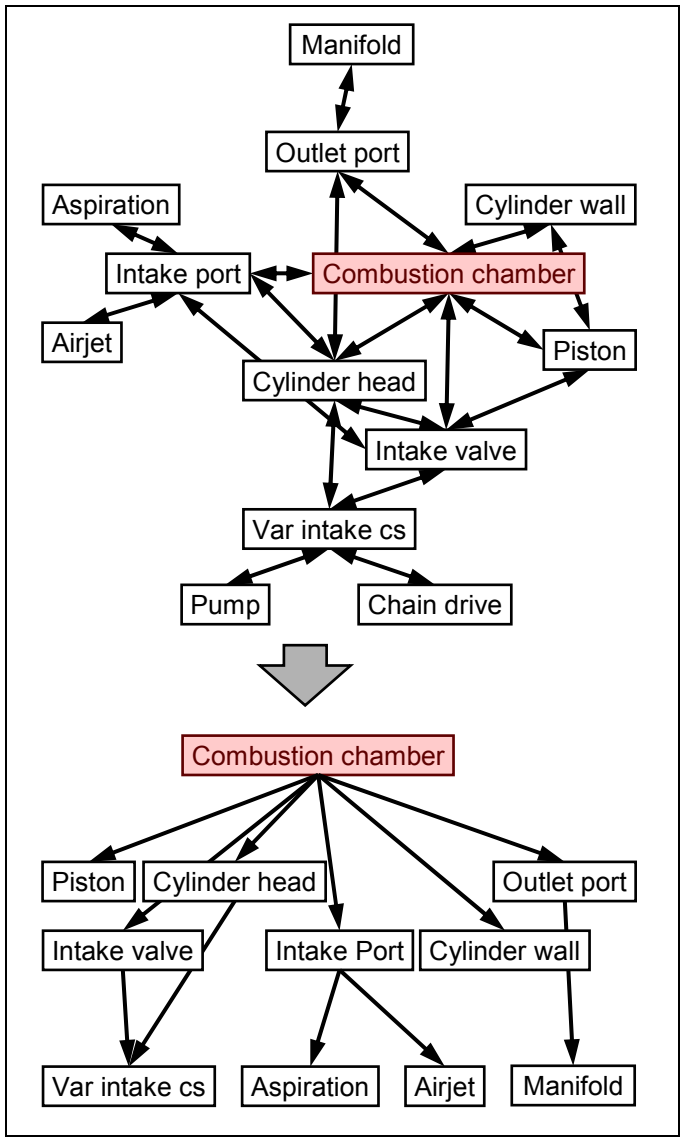
7.4 Methods for the construction of a structure manual

In the following, five methods for the construction of a structure manual are listed in alphabetical order. These methods make use of basic analysis criteria, which permit nodes, edges, subsets, and graphs to be characterized and are specified in the Preceding Sections 7.1 to 7.3. Relevant network elements are highlighted in color in the graphical examples.

7.4.1 Feed-forward analysis

Definition

The feed-forward analysis provides possible chains of change impact that originate from the adaptation of one specific node. This represents an extension to the impact check list, because both the direct surroundings of a specific node and the sequences of outgoing dependencies are considered.



Significance in static and dynamic networks

The feed-forward analysis indicates probable impact propagation due to existing dependency chains.

Challenge of structure improvement

It is nearly impossible to depict and consider all possible dependency chains because of their large quantity in many practical applications. Further constraints, for example, important nodes that have to be included in dependency chains considered, are required to obtain a useful selection.

Potential

The feed-forward-analysis allows the probable consequences of adaptations that may normally remain hidden due to far reaching indirect dependency effects to be estimated.

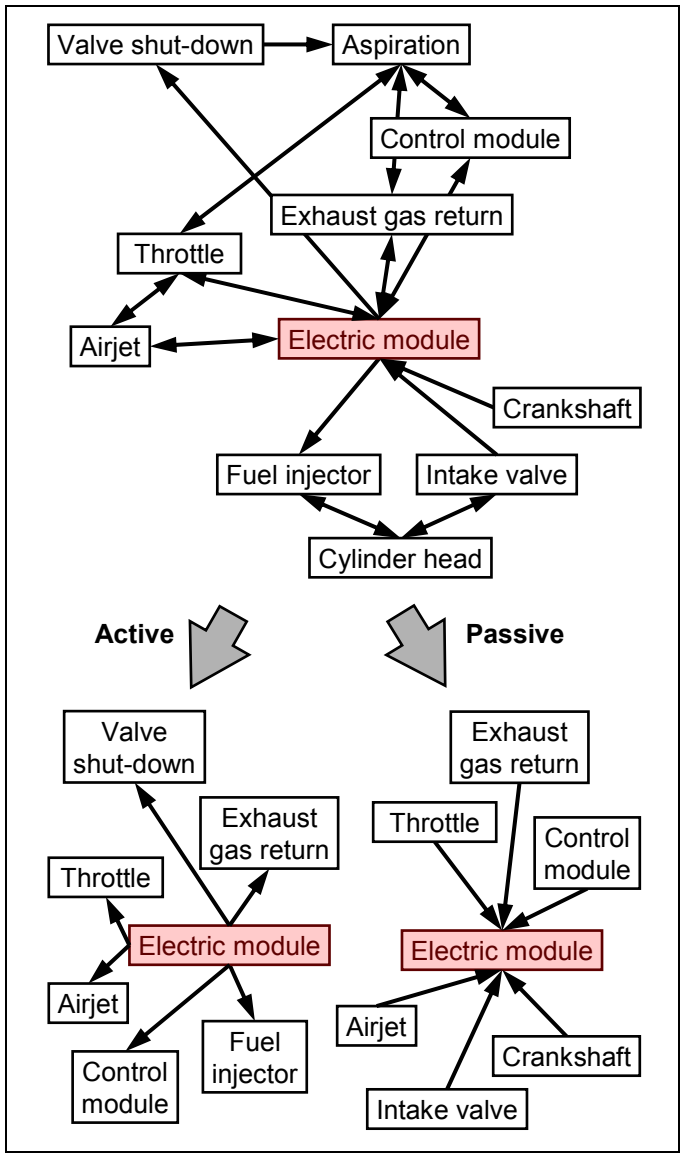
Literature

[HUB 1994, P. 114FF]

7.4.2 Impact check list

Definition

The impact check list provides information about the nodes that are directly linked to one specific node in question. Depending on the use case, the impact check list can comprise either all directly linked nodes or only nodes connected by outgoing or incoming dependencies.



Significance in static networks

The impact check list serves to identify probable direct impact from or to other nodes of a structure.

Significance in dynamic networks

The impact check list indicates tasks that have to be arranged preceding or subsequent to the specifically observed task because of the required input.

Challenge of structure improvement

It is nearly impossible to depict and consider all possible impact check lists because of their large quantity in many practical applications. Further constraints, for example, important nodes that have to be considered, are required to obtain a useful selection.

Potential

The provision of impact check lists supports the systematic step-by-step evaluation of possible impact propagation resulting from the adaptation of a specific system node. If an adaptation has to be executed, product developers can sequentially evaluate the probable occurrence of change impacts to further nodes; their systematic provision guarantees that all possible impacts are considered.

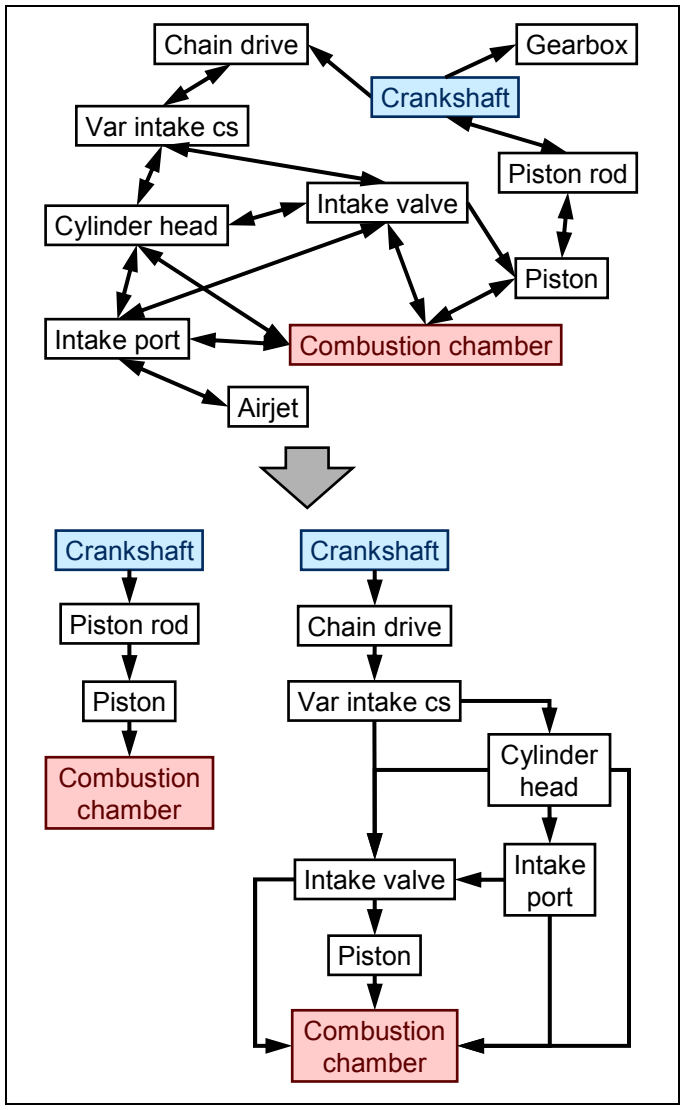
Literature

[DAENZER & HUBER 1999, P. 550FF]

7.4.3 Mine seeking

Definition

Mine seeking represents an application of a feed-forward-analysis. Specific chains of change impact are pointed out, which reach from nodes that are to be adapted to nodes of major significance in the structure.



Significance in static networks

"Structure mines" represent nodes that possess a specific structural embedding in the structure, for example, busses that extensively spread changes. The identified paths indicate probable impact emerging from nodes that have to be adapted to these "structure mines". These impacts have to be avoided.

Significance in dynamic networks

Identified paths can show sequences of activities that end in activities of major importance and require input by the originating node.

Challenge of structure improvement

Destination nodes of impact chains that are unsuitable for adaption must be identified. Often a multitude of dependency paths exist from user-defined nodes to "structure mines". It is nearly impossible to consider all of them, and constraints are required to obtain useful selections.

Potential

"Mine seeking" helps developers identify planned adaptations that would cause (indirect) impact to system nodes, which can provoke enormous adaptation effort if they are affected. Thus, system changes that appear to be unproblematic may be identified as harmful because of change propagation effects.

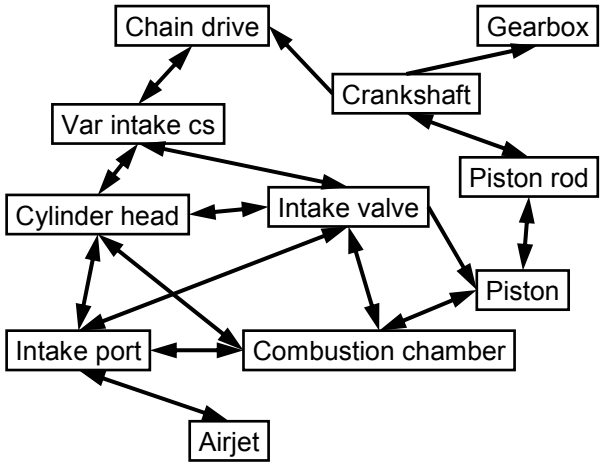
Literature

[HUB 1994, P. 126FF]

7.4.4 Structural ABC-analysis

Definition

The structural ABC-analysis helps identify nodes and edges that are most involved in specific structural subsets, such as feedback loops, hierarchies, or clusters. The results of the analysis can be presented in list form that shows nodes or edges and their quantity of appearance in specific structural constellations in descending sequence. Enhanced representations apply portfolios, for example, an influence portfolio that comprises the active and passive sums of nodes on the two axes.



Edges	Number of feedback loops
Chain drive – Var intake cs	17
Piston – Piston rod	17
Piston rod – Crankshaft	17
Var intake cs – Cylinder head	16
…	…
…	…
Intake valve – Var intake cs	6

Significance in static and dynamic networks

The structural ABC-analysis indicates the nodes and edges that contribute most to the occurrence of specific structural constellations.

Challenge of structure improvement

Typically, algorithmic support is required and computations can be extensive in practical applications.

Potential

The structural ABC-analysis provides knowledge that enables developers to influence effectively the quantity of specific system characteristics. If developers want to reduce the number of feedback loops, for example, this can often be achieved by eliminating one of the edges that occur in a large quantity of feedback loops.

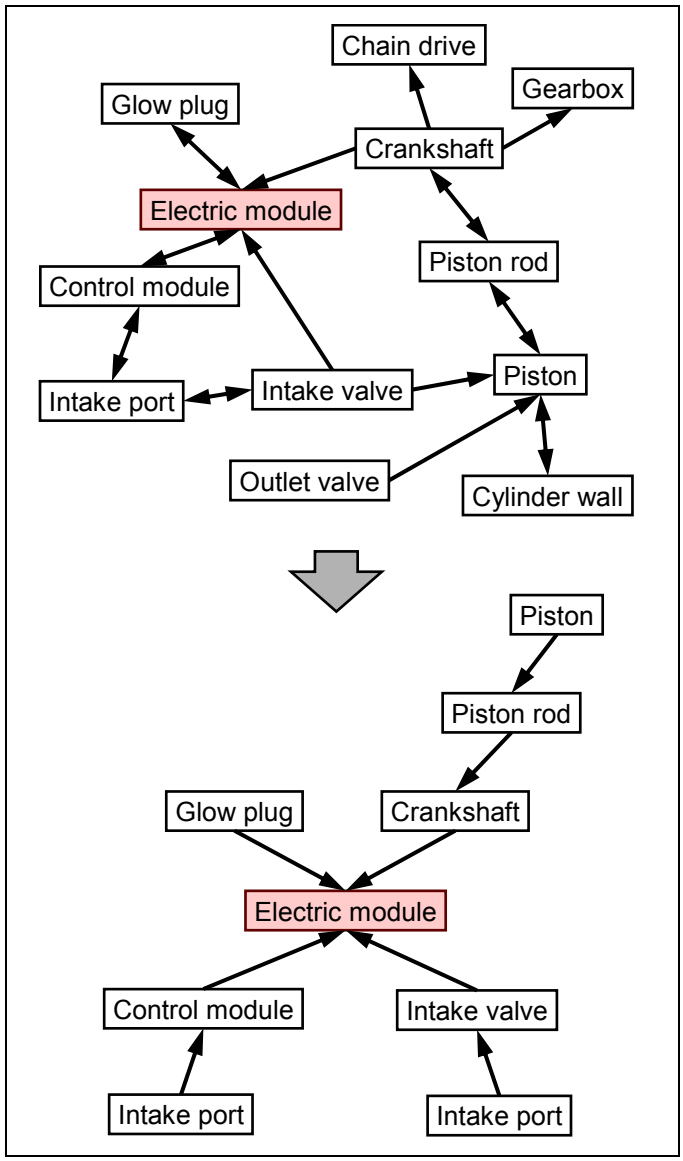
Literature

[HUB 1994, P. 114FF]

7.4.5 Trace-back analysis

Definition

The trace-back analysis depicts possible chains of change impacts starting from specifically observed nodes that show an undesired effect. From these nodes, dependency paths are traced back to the nodes that initiated the undesired effect by, for example, an inappropriate adaptation. The originating nodes can be identified by examining the passive dependency chains belonging to the node in question.



Significance in static and dynamic networks

The trace-back analysis allows the identification of probable and undesired impact propagation due to existing dependency chains.

Challenge of structure improvement

In complex systems not all trace-back routes can be systematically observed, and reasonable constraints must be defined for obtaining a useful selection.

Potential

The systematic tracing of dependency chains starting from the last node to be influenced enables developers to evaluate effectively if the undesired effect emerged from a specific dependency path. If a node in the direct area of the last node affected did not change in the specific scenario, all dependency chains passing through this node are irrelevant for further analyses. Thus, analyses can be kept to a minimum.

Literature

[HUB 1994, P. 126FF]

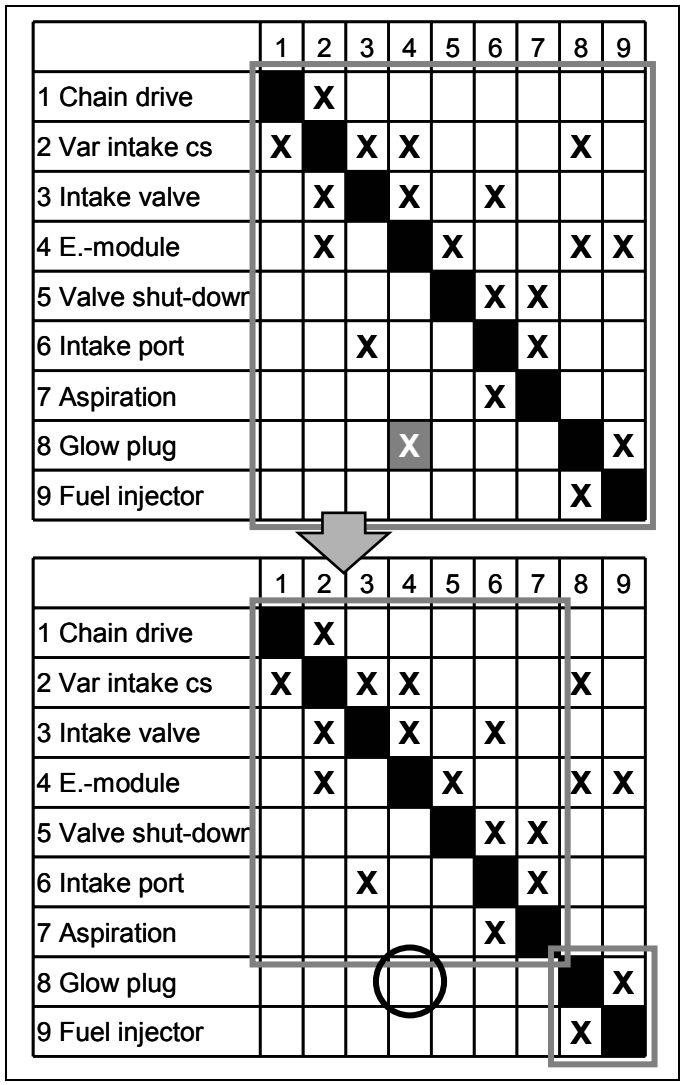
7.5 Methods for optimizing system structures

In the following, two methods for optimizing system structures are listed. These methods apply basic analysis criteria, which allow nodes, edges, subsets, and graphs (explained in the preceding Sections 7.1 to 7.3) to be characterized. The structural ABC-analysis also represents a useful optimization method but is already explained in the preceding Section 7.4 in the context of the structure manual construction. In Section 7.5.2, information about the evolutionary algorithm is presented, whose functional principle is introduced in Section 3.6.2. An application scenario for the structural optimization of a packing machine is shown in Section 4.1.1. Section 7.5.2 provides complementary parameters and quantified dependency values of this specific algorithm application.

7.5.1 Tearing

Definition

Tearing is based on the elimination of edges identified by realignment of a DSM. Tearing requires the application of a partitioning algorithm, which tries to arrange all edges at one side of the matrix diagonal. If this can not be realized (e.g., due to existing feedback loops), partitioning tries to arrange the edges as close as possible to the diagonal. Tearing then tries to identify edges whose elimination can significantly increase the partitioning result.



Significance in static networks

If dependencies that are located at a distance from the matrix diagonal can be avoided, the modularized design of the structure in question can be improved.

Significance in dynamic networks

Edges that are located at a distance from the matrix diagonal often cause long and time-consuming iteration chains. The tearing method selects such edges and proposes their elimination for shortening or even avoiding iterative steps.

Challenge of structure improvement

The selection of edges for tearing depends on the quality of the preceding application of partitioning. Often several different matrix alignments can result, as many structural constellations do not allow one single optimum to be specified. Therefore, it is difficult to guarantee an appropriate selection of edges for improving the structure by their elimination.

Potential

Tearing can help achieve improved modular design and a shorter length of iterations.

Literature

[DSM Internet Page 2007]

7.5.2 Structural optimization with an evolutionary computation

Figure 7-1 depicts the structure of change impacts between the main modules of a packing machine (see the description of the case study in Section 4.1.1) as a distance matrix. That means that values in the matrix cells represent the length of the shortest dependency path between two nodes. In the columns at the right side and the rows below the distance matrix, indexes are computed. Their meanings are listed below. In the columns at the right side of the matrix, active indexes are represented, and the rows below the matrix represent passive indexes.

OL: Overall length;
Sum of all (active or passive) paths from or to a node; the active overall length is depicted as line total, the passive overall length is depicted as column total;

RN: Reachable nodes;
Quantity of nodes that are reached by dependency paths from the specific node or that can reach the node in question by dependency paths;

AT: Attainability;
A detailed explication of this basic analysis criterion can be seen in Section 3.5.3;

CL: Closeness;
A detailed explication of this basic analysis criterion can be seen in Section 3.5.3;

AF: Active fitness;
Value computed according to Equation 3-17 in Section 3.6.2;

PF: Passive fitness;
Value computed according to Equation 3-18 in Section 3.6.2;

The overall fitness value for the initial structure is computed according to Equation 3-19 (Section 3.6.2) and is depicted in the lower right corner of Figure 7-1.

Figure 7-2 depicts the distance matrix together with its indexes for an optimized structure of the packing machine, where two dependencies have been removed by application of the GA. The settings chosen for the application of the genetic algorithm are:

Number of iterations: 1000

Ratio of active fitness to passive fitness: $w_a/w_p = 1$;

Ratio of attainability to closeness: $v_a/v_p = 1$;

The overall fitness value for the optimized structure is computed according to Equation 3-19 (Section 3.6.2) and is depicted in the lower right corner of Figure 7-2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	OL	RN	AT	CL	AF
A		1	3	3	3	3	3	3	2	3	1	2	1	2	3	3	3	3	3	1	2	4	1	2	1	56	24	10	23	17
B	1		4	4	4	4	4	4	3	4	2	3	1	3	4	4	4	4	4	1	3	5	2	1	1	74	24	10	31	20
C	1	2		1	4	1	4	4	2	1	2	3	2	3	4	4	4	4	4	2	3	5	2	2	2	66	24	10	28	19
D	1	1	1		4	2	4	4	2	1	2	3	2	3	4	4	4	4	4	2	3	5	2	2	2	66	24	10	28	19
E	2	2	1	1		1	2	2	2	2	2	3	2	2	2	1	2	2	1	3	1	1	2	2	3	44	24	10	18	14
F																										0	0	0	0	0
G	1	2	3	3	3	3		1	2	3	1	1	2	2	1	1	3	3	3	2	2	4	2	3	1	52	24	10	22	16
H	1	2	4	4	4	4	1		3	4	2	1	2	3	1	1	4	4	4	2	3	5	2	3	1	65	24	10	27	19
I	1	2	2	1	3	3	3	3		2	1	2	2	2	3	3	3	3	3	2	2	4	2	3	2	57	24	10	24	17
J	1	1	1	1	4	2	4	4	1		2	3	2	3	4	4	4	4	4	2	3	5	2	1	2	64	24	10	27	18
K	1	1	2	2	2	2	2	2	1	2		1	1	1	2	2	2	2	2	1	1	3	2	2	2	41	24	10	17	14
L	1	2	3	3	3	3	2	1	2	3	1		2	2	1	1	3	3	3	2	2	4	2	3	2	54	24	10	23	16
M	1	1	4	4	4	4	4	4	3	4	2	3		3	4	4	4	4	4	1	3	5	2	2	2	76	24	10	32	21
N	2	2	3	3	3	3	3	2	2	3	1	1	2		2	2	3	3	3	2	2	4	3	3	3	60	24	10	25	18
O	1	2	4	4	4	4	1	1	3	4	2	1	2	3		1	4	4	4	2	3	5	2	3	2	66	24	10	28	19
P	2	3	5	5	5	5	2	1	4	5	3	2	3	4	2		5	5	5	3	4	6	3	4	2	88	24	10	37	23
Q	2	3	4	4	4	4	1	2	3	4	2	2	3	3	2	2		4	4	3	3	5	3	4	2	73	24	10	30	20
R	3	4	5	5	5	5	2	2	4	5	3	3	4	4	3	1	1		5	4	4	6	4	5	3	90	24	10	38	24
S	1	1	3	2	4	1	4	4	1	3	2	3	2	3	4	4	4	4		2	3	1	2	2	2	62	24	10	26	18
T	1	1	4	4	4	4	4	4	3	4	2	3	1	3	4	4	4	4	4		3	5	2	2	2	76	24	10	32	21
U	1	2	1	1	1	1	1	2	2	1	1	2	1	1	1	1	1	1	1	2		2	1	1	2	31	24	10	13	11
V	1	2	4	4	4	4	4	4	3	4	2	3	2	3	4	4	4	4	4	2	3		2	3	2	76	24	10	32	21
W	1	2	4	4	4	4	4	4	3	4	2	3	2	3	4	4	4	4	4	2	3	5		1	2	77	24	10	32	21
X	2	3	5	5	5	5	5	5	4	5	3	4	3	4	5	5	5	5	5	3	4	6	1		3	100	24	10	42	26
Y	1	1	4	4	4	4	4	4	3	4	2	3	2	3	4	4	4	4	4	2	3	5	2	2		77	24	10	32	21
																														18
OL	30	43	74	72	85	76	68	67	58	75	43	55	46	63	68	64	79	82	82	48	63	100	48	56	46					
RN	23	23	23	23	23	24	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23					
AT	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10					
CL	13	18	31	30	35	32	28	28	24	31	18	23	19	26	28	27	33	34	34	20	26	42	20	23	19					
PF	11	14	20	20	23	21	19	19	17	20	14	16	14	18	19	18	21	22	22	15	18	26	15	16	14	18				

Overall fitness 18,06

Figure 7-1 Distance matrix of the initial dependency structure matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	OL	RN	AT	CL	AF
A		1											1							1			1	2	1	7	6	3	3	3
B	1												1							1			2	1	1	7	6	3	3	3
C	1	2		1		1	6	5	2	1	3	4	2	4	5	5				2			2	2	2	50	18	8	21	14
D	1	1	1			2	6	5	2	1	3	4	2	4	5	5				2			2	2	2	50	18	8	21	14
E	2	2	1	1		1	2	2	2	2	2	3	2	2	2	1	2	2	1	3	1	1	2	2	3	44	24	10	18	14
F																										0	0	0	0	0
G	1	2	4	3		5		1	2	4	1	1	2	2	1	1				2			2	3	1	38	18	8	16	12
H	1	2	5	4		6	1		3	5	2	1	2	3	1	1				2			2	3	1	45	18	8	19	13
I	1	2	2	1		3	4	3		2	1	2	2	2	3	3				2			2	3	2	40	18	8	17	12
J	1	1	1	1		2	5	4	1		2	3	2	3	4	4				2			2	1	2	41	18	8	17	12
K	1	1	3	2		4	3	2	1	3		1	1	1	2	2				1			2	2	2	34	18	8	14	11
L	1	2	4	3		5	2	1	2	4	1		2	2	1	1				2			2	3	2	40	18	8	17	12
M	1	1																		1			2	2	2	9	6	3	4	3
N	2	2	4	3		5	3	2	2	4	1	1	2		2	2				2			3	3	3	46	18	8	19	13
O	1	2	5	4		6	1	1	3	5	2	1	2	3		1				2			2	3	2	46	18	8	19	13
P	2	3	6	5		7	2	1	4	6	3	2	3	4	2					3			3	4	2	62	18	8	26	17
Q	2	3	5	4		6	1	2	3	5	2	2	3	3	2	2				3			3	4	2	57	19	8	24	16
R	3	4	6	5		7	2	2	4	6	3	3	4	4	3	1	1			4			4	5	3	74	20	8	31	20
S	1	1	3	2		1	5	4	1	3	2	3	2	3	4	4				2		1	2	2	2	48	20	8	20	14
T	1	1											1										2	2	2	9	6	3	4	3
U	1	2	1	1	1	1	1	2	2	1	1	2	1	1	1	1	1	1	1	2		2	1	1	2	31	24	10	13	11
V	1	2											2							2			2	3	2	14	7	3	6	4
W	1	2											2							2				1	2	10	6	3	4	3
X	2	3											3							3			1		3	15	6	3	6	4
Y	1	1											2							2			2	2		10	6	3	4	3
																														10

OL	30	43	51	40	1	62	44	37	34	52	29	33	46	41	38	34	4	3	2	48	1	4	48	56	46					
RN	23	23	15	15	1	16	15	15	15	15	15	15	23	15	15	15	3	2	2	23	1	3	23	23	23					
AT	10	10	6	6	0	7	6	6	6	6	6	6	10	6	6	6	1	1	1	10	0	1	10	10	10					
CL	13	18	21	17	0	26	18	15	14	22	12	14	19	17	16	14	2	1	1	20	0	2	20	23	19					

PF	11	14	14	11	0	16	12	11	10	14	9	10	14	12	11	10	1	1	1	15	0	1	15	16	14	10

Overall fitness
9,84

Figure 7-2 Distance matrix of the optimized dependency structure matrix

8. List of dissertations

Lehrstuhl für Produktentwicklung
Technische Universität München,
Boltzmannstraße 15
85748 Garching

Dissertations under supervision of

- Prof. Dr.-Ing. W. Rodenacker,
- Prof. Dr.-Ing. K. Ehrlenspiel
- 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: TU, Diss. 2007.

D95 PONN, J.:
Situative Unterstützung der methodischen Konzeptentwicklung technischer Produkte.
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.
TU München: 2007. (als Dissertation eingereicht)

D98 FELGEN, L.:
Systemorientierte Qualitätssicherung für mechatronische Produkte.
München: TU, Diss. 2007.

D99 GRIEB, J.:
Auswahl von Werkzeugen und Methoden für verteilte Produktentwicklungsprozesse.
TU München: 2007. (als Dissertation eingereicht)