Information visualization for the Structural Complexity Management Approach

Maik Maurer, Thomas Braun
Teseon GmbH
Parkring 4, 85748 Garching (Germany)
Tel: +49 89 3074815-0,
maik.maurer@teseon.com

Udo Lindemann
Technische Univiversität München
Boltzmannstr. 15, 85748 Garching
(Germany)
udo.lindemann@pe.mw.tum.de

Abstract. The handling of complexity poses an important challenge and a success factor for product design. A considerable percentage of complexity results from dependencies between system elements – as adaptations to single system elements can cause far-reaching consequences. The Structural Complexity Management (SCM) approach provides a five-step procedure that supports users in the identification, acquisition, analysis and optimization of system dependencies. The approach covers the handling of multiple domains (e.g. components, functions and people).Additionally the application of the SCM approach requires intensive support by information visualization, e.g. matrix and graph representations. However, one single technique does not satisfy all demands. Each phase of the SCM approach possesses individual requirements on the visual support. Whereas many specific techniques exist for visualizing systems and their dependencies (e.g. mind map or DSM), a suitable combination that supports the entire SCM approach is not available. For this reason, practical applications of the SCM approach are only restricted to small systems so far. In this contribution, we classify available techniques of information visualization and match them with the requirements of the SCM approach. Then we assemble a comprehensive model of visualization techniques which will provide the implementation basis for an appropriate software support.

Introduction

The problem. The management of complex technical products poses an important challenge in engineering. Among other criteria, in the first instance the amount of dependencies determines a system's complexity and therefore several methodical approaches address the handling of dependencies. These approaches apply possibilities of information visualization in order to provide an intuitive system understanding and to allow user interaction.
The Structural Complexity Management (SCM) approach (Lindemann et al., 2008) supports the handling of multiple-domain systems, i.e. systems containing several aspects (e.g. components, functions, signals) and dependency types (e.g. flows of forces, change impact, information flow). This approach covers the entire process from the initial system definition until the implementation of improved system management and design. The single steps need to be supported by techniques of information visualization (e.g. the DSM); however, a comprehensive model for maintaining the requirements on visualization, mediation, and analysis of the approach does not exist. For this reason, practical applications of the SCM approach are restricted to small systems so far.

The objective. We exemplify suitable forms of information visualization and match them with the requirements of the SCM approach. This results in an assembly of visualization techniques and their required interfaces that support the demonstration of all process steps. This assembly represents the basis for a tool implementation that allows the practical application of the SCM approach. In this way the management of complex design tasks can be improved significantly.

Complexity management as a multiple-domain task. Concerning modern automobiles, a multitude of technical dependencies emerged due to increased integration of mechatronical components. Consequently, not only product complexity but also process complexity increased, to some extent because mechatronics implied an intense division of labor and an increased need for development coordination. Additionally, the increase of product functionalities ended up in the necessity of intensifying cooperation of internal enterprise departments and external suppliers (Kusiak, 1999). As a result organizational and communication flows also become more complex. Considering only the isolated aspects of complexity (i.e. single-domain systems) is often misleading. Actions derived for improving the complexity of specific system extractions can contradict requirements that have not been considered. Consequently, complexity management often means to cope with several system aspects simultaneously.

Consideration of system structures. Every system containing of at least two or more parts that interact with each other possesses an underlying structure (Boardman & Sauser, 2006). Knowledge about this structure allows conclusions about the system itself and results in improved understanding. In technically demanding products, multiple dependencies exist between integrated components and therefore considerably complicate the design of a specific component. The determination of consequences that can result from single adaptations requires information about the internal product dependencies. Knowledge about this structure of connectivity, for example, permits developers to identify required actions that are related to a specific customer request.

Visualization of complex system dependencies

Many approaches allow representing graphically and interacting with dependencies in complex systems. For example, several hundred projects are documented at visualcomplexity (`http://www.visual complexity.com`). These projects refer to a large variety of application fields and mainly apply graph representations. Some authors also mention the various fields of matrix application for handling complex system dependencies (e.g. Yassine, 2004; Browning, 2001; Danilovic & Browning, 2004). The gestalt laws depict the fundamentals of these graph and matrix representations (Ware, 2004). In the following, we provide a general classification of matrix- and graph-based approaches that constitute the basic set of methods available for supporting the application of the SCM approach.

Matrix-based approaches

Matrix-based visualizations for representing system dependencies can be classified by the quantity of the types of elements involved (Figure 1), whereas some approaches focus on the representation (and analysis) in between elements of the same type (e.g., dependencies within product components), others consider linkages between two types (e.g., dependencies between customer requirements and product functions). In the first approach, the related matrices can be defined as intra-domain. The label “domain” describes the type of element, e.g. “product components”. In the second approach, matrices are referred to as inter-domain matrices. More comprehensive approaches take into account even more domains by the sequential combination of intra- and inter-domain matrices. Finally, the Multiple-Domain Matrices (MDMs) apply computations between combined matrices and can provide specific views on a modeled system.

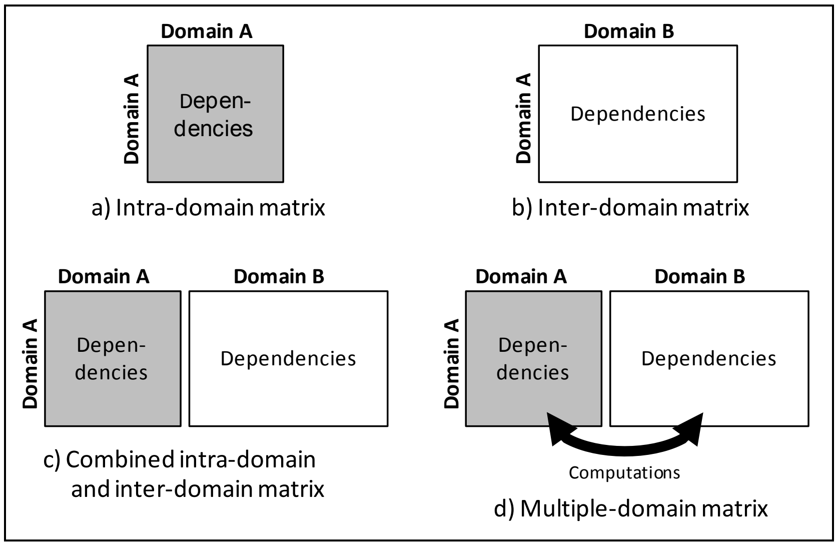


Figure 1. Classification of matrix-based models (Lindemann et al., 2008)

Intra-domain matrices. A popular method using intra-domain matrices is the Dependency Structure Matrix (DSM). According to (Browning, 2001) a DSM is a matrix with an equal number of rows and columns. It provides systematic mapping of elements (all belonging to one domain only) and their relationships. 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 use of DSMs has been extended to many types of system and design analysis, e.g., project planning, project management, and organization design (Brady, 2002). In addition to the DSM, many further applications of intra-domain matrices exist in product design, e.g., compatibility matrices.

Inter-domain matrices link elements of two different domains and are widely used in design methodology, e.g. the “Cause and Effect Matrix” (Allen, 2006). In 2001, Danilovic & Börjesson settles on the term “Domain Mapping Matrices” (DMMs) for an enhancement of the DSM methodology to inter-domain matrices. The authors presented studies on linkages between product architecture and organization as well as between systems and organization.

Combined intra- and inter-domain matrices represent a combinatorial enhancement of the matrix types mentioned before. For example, the House of Quality (integral part of the method of Quality Function Deployment) combines inter- and intra-domain matrices to capture several system aspects (domains) simultaneously (Akao, 1992). The basic form of the House of Quality contains of one intra-domain matrix and three inter-domain matrices linking customer requirements, technical requirements, criteria, and technical evaluation together.

Multiple-Domain Matrices (MDMs) differ from the combined application of intra- and inter-domain matrices by computations within the considered matrices. In 2008, Lindemann et al. provided a detailed definition of the configuration of a MDM. Figure 2 shows the layout of a MDM together with an associated graph representation. The MDM consists of intra- and inter-domain matrices that allow the modeling of dependencies within and between elements belonging to several domains. In addition, specific system views can be derived by computational analysis of indirect dependencies in the MDM (see also Figure 8 and related explications).

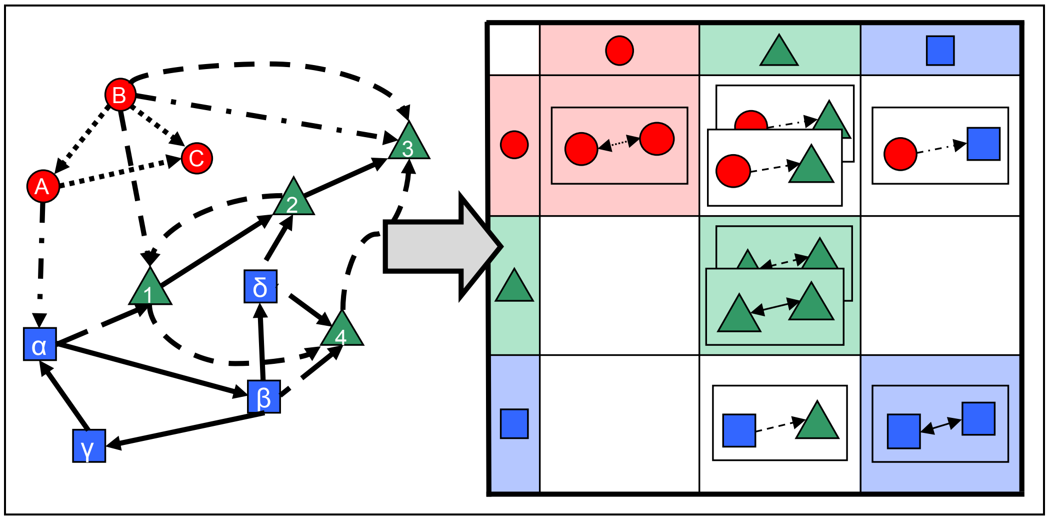


Figure 2. Multiple-Domain Matrix and associated graph representation

Graph-based approaches

Generally, graph theory and graph drawings must be clearly separated. Graph theory represents the mathematics of networks, i.e. system elements and their connections. Thus, graph theory provides the fundamentals for many methods applied in product design, e.g., project scheduling by the critical path method (Gross & Yellen, 2006). Systems can be characterized by their implied substructures (trees, cycles, etc.) or their structural attributes like connectivity, coloring, or planarity of considered graphs, just to mention a few, formed by elements and dependencies (Gross & Yellen, 2006; Bollobás, 1990). Graph drawings, however, mean the depiction of networks mostly by nodes and edges. Thus, graph drawings (as well as matrices) represent a possibility to depict the mathematical formulations of graph theory (Gross & Yellen, 2006).

Simple graphs and digraphs. Simple graphs can depict a general linking of elements; digraphs can even model information about a flow direction and not merely the connectivity of two elements. Due to their generic principle of information representation digraphs can be found in many applications, e.g. in modeling process flows or cause-and-effect chains.

Graphs applying ordering schemes provide enhanced possibilities of intuitive representation and interaction with complex systems. An ordering scheme helps users to identify relevant elements, as attributes of elements directly relate to their position in a graph drawing. For example, the time attribute can be applied for aligning process tasks along a time line. Even if a system contains many elements, their attribute-based classification makes them easy to detect in the visualization. Another common ordering scheme represents the degree of system decomposition of elements. Often, this scheme gets applied to the depiction of a bill of material and results in a hierarchical alignment of system components (product tree).

Force-directed graphs apply an ordering scheme, which does not base on attributes of elements. In fact, force-directed graphs use the dependencies (edges) of a graph for creating the positioning of all system elements in the visual alignment. According to Di Battista et al., 1999 “force directed algorithms are intuitive methods for creating straight-line drawings of undirected graphs. They are quite popular because their basic versions are easy to understand”. The principle of force-directed graphs can be seen in Figure 3. Depiction a) “shows a graph where vertices have been replaced with electrically charged particles that repel each other and edges have been replaced with springs that connect the particles. An equilibrium configuration, where the sum of the forces on each particle is zero, is illustrated in [depiction b) of Figure 3]. This configuration can be interpreted as a straight-line drawing of the graph, as in [depiction c) of Figure 3]" (Di Battista et al., 1999).

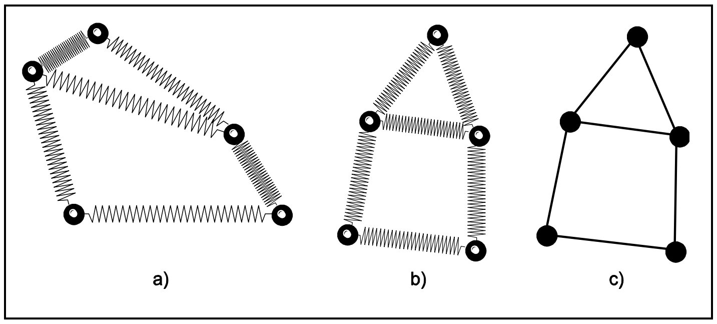


Figure 3. Compilation of a force-directed graph (according to Di Battista et al., 1999)

Multiple-domain graphs. Several applications require the simultaneous depiction of elements belonging to different domains (`http://www.visualcomplexity.com`). All kinds of graph drawings can be used for such visualizations. However, clarity and possibilities of user interaction decrease with additionally integrated domains. Figure 4 shows a digraph (without ordering scheme) containing elements belonging to four different domains.

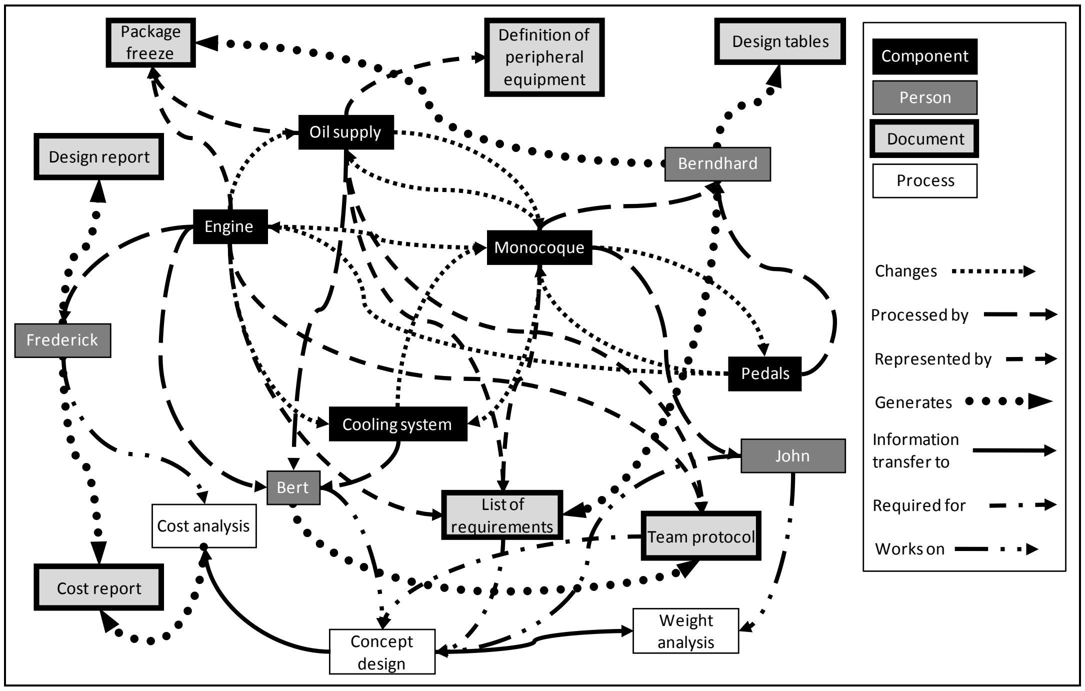


Figure 4. Multiple-domain graph

An attribute-based ordering scheme could be applied to the visualization, if all elements would share a common attribute. However, this can hardly be found in typical systems. In Figure 4, for example, the time attribute belongs to elements of the process domain but not to elements of the people domain. Generally, two ordering schemes (e.g. a horizontal and vertical axis) could be applied to a multiple-domain graph simultaneously. But this can result in complicated and equivocal element positions and decreases possibilities of intuitive understanding. The force-directed ordering scheme could also be applied to multiple-domain graphs; but dependencies need to possess identical meaning. This is not the case in typical multiple-domain graphs.

Procedure of the Structural Complexity Management Approach

The Structural Complexity Management (SCM) approach supports the handling of multiple-domain systems (Lindemann et al., 2008) containing numerous dependencies between system elements. The approach contains five steps (Figure 5), which will be introduced in the following. The suitable information visualization represents a key factor for the successful application of the approach. For this reason, we try to identify the appropriate visualization techniques required to fulfill the specific tasks of each step of the approach. Next, these techniques are combined in a comprehensive model that provides the basis for a software implementation.

Introduction of the SCM approach. Here, the comprehensive approach will be shortly introduced. A detailed description and application scenarios can be taken from (Lindemann et al., 2008). The initial situation 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 due to unknown or undesired side-effects. A specific adaptation request can seem not to be complex, but system dependencies may lead to numerous subsequent adaptations (change propagation). This can result in time and resource shortages when attempting to accommodate the desired adaptation. A design problem can occur in the new development of a system, e.g., a product. If product development includes a pro-active structure development, design improvements can be realized; i.e., less iteration occurs in comparison to passively emerged structures and change propagations can be locally restricted.
During the initial system definition the general extent of the considered system has to be determined. That includes the identification of required domains, the determination of system elements and the dependency types that link between the domains.
In the step of information acquisition direct dependencies between system elements are identified and documented in the system model. Such dependencies can, e.g., be acquired from data bases, modeling tools applied in the design process, or by interviews with system experts. The main challenge of information acquisition is to guarantee a high data quality.
If the direct dependencies between system elements are on hand, the deduction of dependencies allows deriving specific subsystems. These subsystems represent focused views on specific system aspects. The deduction of dependencies provides intra-domain networks that allow subsequent system analysis and interpretation. An example can clarify this: In the step of information acquisition the direct dependencies between product components have been acquired (change propagation) as well as the dependencies between product components and designers (responsibility of designers for components). Now, the application of the deduction of dependencies can provide a network between product designers, who are indirectly linked, because of their work on related hardware components. In fact, the designers are linked indirectly in the system and the resulting network represents a specific view on organizational aspects in the product design.
During the step of structure analysis the system's characteristics are identified by applying methods of the algorithmic graph theory (Kusiak, 1999). Significant structural constellations, e.g.

feedback loops or clusters can be detected, which are implied in the system structure.
In the final step of product design application acquired knowledge about significant constellations is used to improve the system design or the system handling. A structure manual, for example, can be created providing a better understanding and awareness of the system. Even better such a structure manual serves as a guideline for the systematic handling of system complexity.

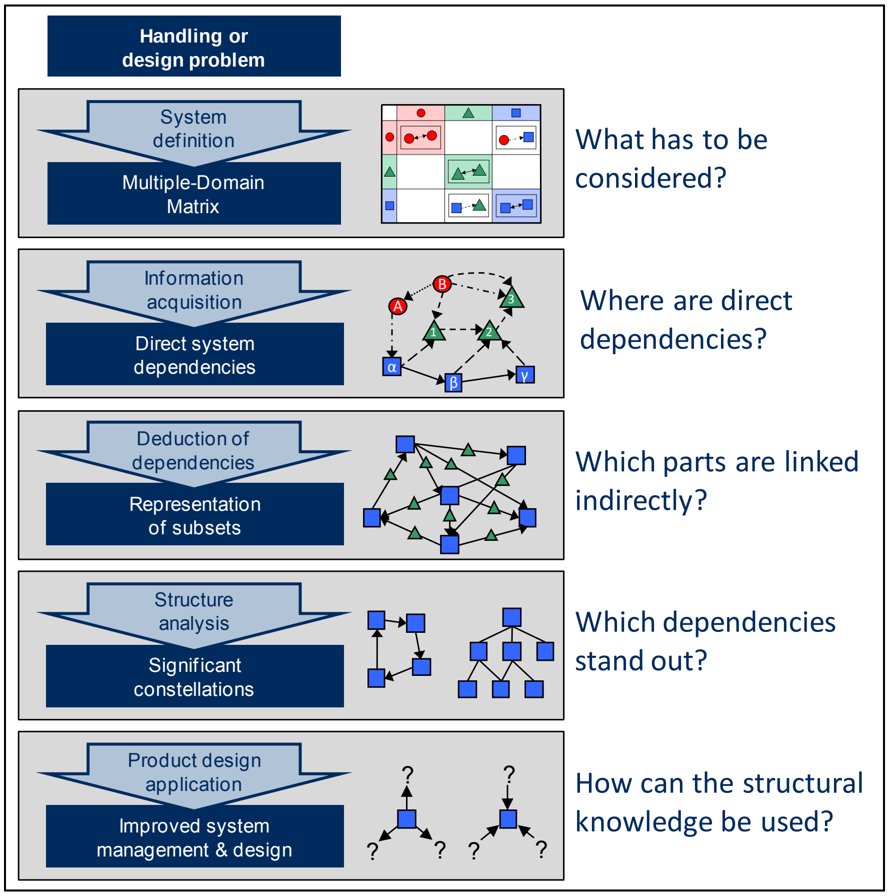
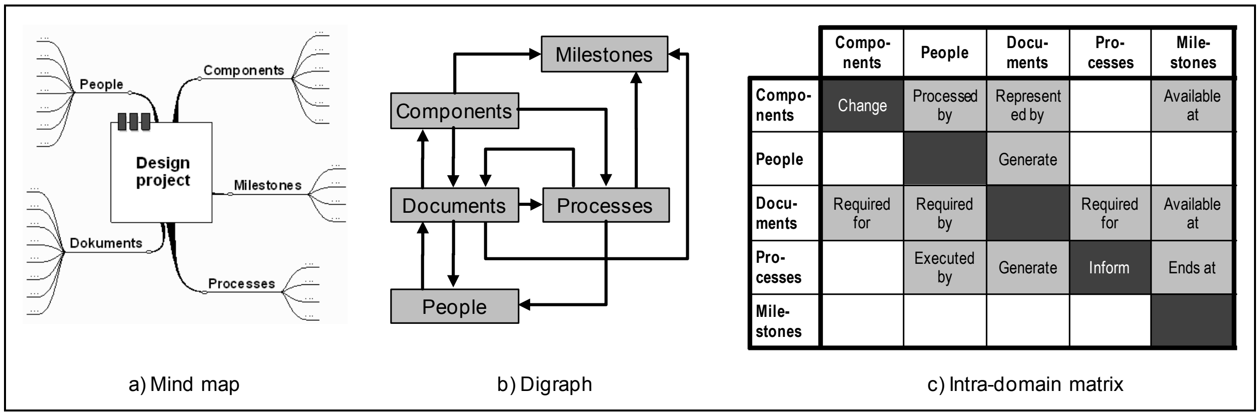

Handling or design problem
System definition
Multiple-Domain Matrix
What has to be considered?
Information acquisition
Direct system dependencies
Where are direct dependencies?
Deduction of dependencies
Representation of subsets
Which parts are linked indirectly?
Structure analysis
Significant constellations
Which dependencies stand out?
Product design application
Improved system management & design
How can the structural knowledge be used?

Figure 5. Structural Complexity Management Approach (Lindemann et al., 2008)

System definition. In this phase relevant domains and dependency types are defined. The optimal system definition comprises the minimum of domains and dependency types required for solving the initial problem (definition of the system borders). The identification of relevant domains starts with collecting system elements, which are classified into groups afterwards. This task can be supported by using mind maps (Buzan, 1996). In a hierarchically aligned collection of elements, the elements on the first level represent the domains for subsequent consideration (Figure 6a).
With the domains on hand, their mutual connectivity has to be modeled. Typically, not all domains are linked directly. If the dependencies between domains are known, efforts for the following information acquisition can be kept at a minimum. However, the identification of dependencies between domains requires another representation than the collection of domains, because cross-linking in mind maps becomes easily confusing. Mind maps are meant for the creation of hierarchical links only. Dependency types between domains can be depicted in a digraph without an ordering scheme (Figure 6b); but this representation can become difficult to handle, if the system contains many domains (because of confusing intersections). In this case an intra-domain

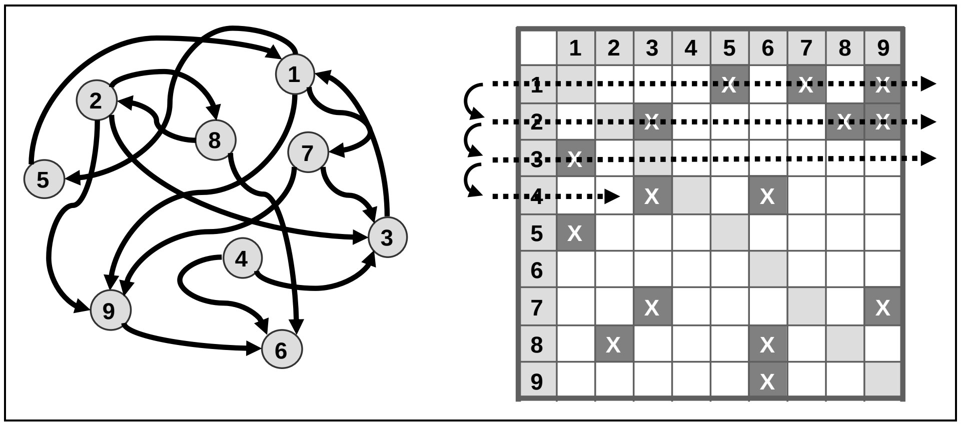
matrix can be applied (Figure 6c), where the domains are noted as column and row heads and dependency types between domains can be noted within the matrix fields. If users investigate all matrix fields sequentially (row by row or column by column) all possible links between domains should be considered.



	Compo-nents	People	Docu-ments	Pro-cesses	Mile-stones
Compo-nents	Change	Processed by	Represented by		Available at
People			Generate		
Docu-ments	Required for	Required by		Required for	Available at
Pro-cesses		Executed by	Generate	Inform	Ends at
Mile-stones					

Figure 6. System definition: Visualizing domains and their dependency types

Information acquisition. In the first step the filled matrix cells from Figure 6c have to be divided into matrix subsets. That means that instead of the domain names the implied system elements are noted in the matrix columns and row heads. Thus, cells on the matrix diagonal become expanded into intra-domain matrices, whereas all other cells become expanded into inter-domain matrices. In the following step dependencies between system elements have to be acquired and documented in the matrix cells. At this point systematic procedure and appropriate information visualization is necessary to achieve completeness and high quality of the acquired network. If dependencies between system elements are captured by drawing connectors in a digraph, the illustration quickly becomes complex (left side in Figure 7). In this case it is difficult to see, if all possible dependencies have been considered not to forget that elements can be difficult to find as soon as networks become larger.
The matrix at the right side of Figure 7 contains the same information as the digraph. For the acquisition of dependencies the matrix can be processed row by row (or column by column). This procedure assures that all element pairings are considered for possible dependencies.



	1	2	3	4	5	6	7	8	9
1					X		X		X
2			X					X	X
3	X								
4			X			X			
5	X								
6									
7			X						X
8		X				X			
9						X			

Figure 7. Suitability of graph and matrix for the systematic information acquisition

Figure 7 clarifies information acquisition for an intra-domain network. Concerning inter-domain networks, an application of a matrix representation gets even more favorable for information acquisition: In a digraph the different element types would result in a more complicated depiction as confusing intersections of dependencies become unavoidable (the total number of considered system elements gets typically larger than in inter-domain networks). In contrary, the matrix depiction and the associated process of information acquisition remains still the same as it was applied to the inter-domain matrices (but each axis contains different system elements).

Deduction of dependencies. The deduction of dependencies represents a mathematical procedure (Lindemann et al., 2008). However, visual support of this process step can facilitate the determination of the target domain and required input data (information sources). Figure 8 clarifies the need for visual support. The graph represents an extraction of a dependency network and contains four system elements belonging to two different domains. The square domain can e.g. represent people, whereas the triangular domain represents product components. The thin arrows indicate the direct dependencies acquired for this system. Thus, geometric dependencies between product components and the responsibility of people for specific components have been identified. Now, the deduction of dependencies allows determining indirect links between elements of one domain by using dependency chains passing by elements of a second domain. In Figure 8 the thick arrow indicates such an indirect link resulting from the three direct ones (thin arrows). For the exemplary use case, this deduced dependency indicates that person d impacts person a by executing a design change to component 2 (which propagates the change to component 1). The deduction of such component-based dependencies between designers is a good example, what a great help it can be to manage development processes.

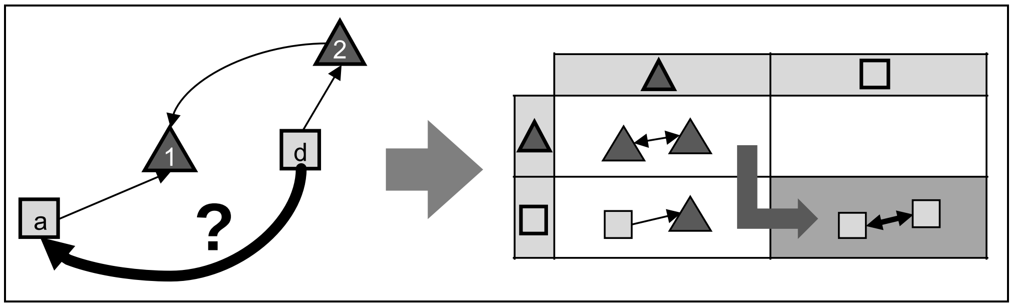


Figure 8. Deduction of dependencies – representation in graph and matrix

Now the indirect connection between both square elements can be easily derived visually in the digraph by following the direct links. However, in complex networks the visual detection of indirect dependencies is mostly impossible. This is why an application of an intra-domain matrix represents a useful alternative for supporting the deduction of dependencies in complex networks. Hereby, the same matrix gets applied as introduced in the step of system definition (Figure 6c). The system domains are implemented as elements in the row heads and column heads of the matrix. The dependency types are documented in the matrix cells. Figure 8 shows the matrix for the two domains mentioned above. If users want to determine dependencies between designers (square domain), these dependencies are modeled in the lower right cell of the matrix (shaded in grey). In other words, the objective of the deduction of dependencies is the network of system elements represented by the lower right matrix cell. The information sources required for the network determination can also be identified in the matrix cells: The upper left and the lower left matrix cell represent the networks of component links and the links between people and components. The example shows that the target domain and the information sources for the deduction of dependencies can be selected from an intra-domain matrix that represents the system domains. Figure 9 depicts the visual matrix support for the deduction of dependencies, as it could be realized in a software implementation. In our example the modeled system contains three domains (components, people and documents). Four types of dependencies have been modeled, each linking between two domains (shown by the dependency meaning written in the matrix cells). Based on this information, five kinds of indirect dependencies can be computed, which result in indirect links between people (indicated by the arrows). At the right side of Figure 9, all required information to compute one specific type of indirect dependencies has been specified: the matrix field in dark grey indicates the target domain (people). The two matrix fields shaded in light grey represent the associated information sources. If the matrix fields are implemented as buttons in a user interface, users can easily determine required indirect dependencies.

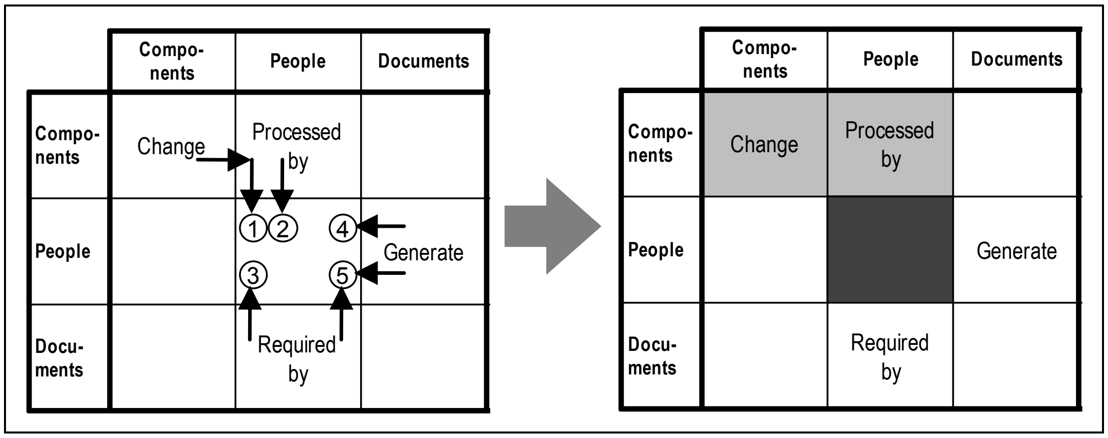
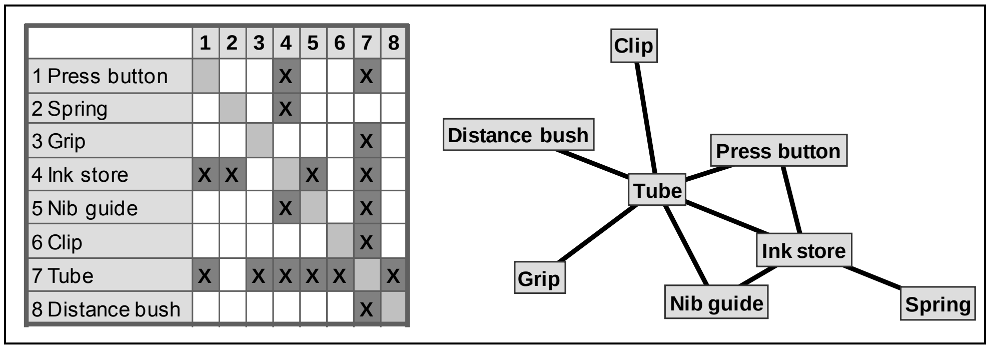


Figure 9. Selecting objectives and information sources for the deduction of dependencies

Structure analysis. Analysis methods can be applied to intra-domain networks (resulting from the deduction of dependencies or directly from information acquisition). Analysis methods help to detect structural characterizations, which can be interpreted later on. Therefore DSM-related research provides appropriate methods (Browning, 2001). Moreover several authors provide descriptions on analysis methods based on graph theory (e.g. Bollobás, 1990; Melnikov et al., 1994). For the application within the SCM approach, the analysis methods can be classified into two groups:

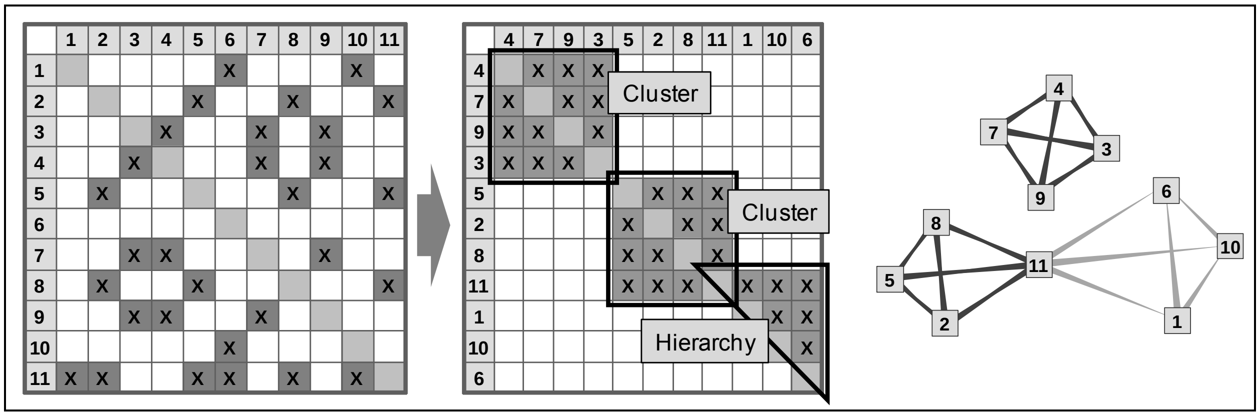
The first group contains analyses that provide direct benefit from the visual examination of a system or by interacting with this visualization. Figure 10, for example, shows the structure of a change propagation based on components as seen at the components of a ballpoint pen. Here, a (non-directional) force-directed graph gets applied. The dependencies between elements 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”). Even if both representations contain the same information, the implied structure is easier to understand by the force-directed graph: The “Tube” represents the core element, as almost all other elements are linked to this. In the graph representation, this element is located in the center, which makes its structural relevance intuitive. Two completely interlinked clusters exist in the structure (“Tube”–“Nib guide”–“Ink store”, “Tube”–“Press button”–“Ink store”). In the graph, both clusters intensified by the fact that they both overlap in the two elements “Tube” and “Ink store” can be easily detected. The matrix at the left side contains the same information as the graph; however, users must be familiar with this representation form for extracting the relevant content.



	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 10. Visual structure analysis applying matrix and graph

Whereas in Figure 10 the graph representation seems to be more appropriate, Figure 11 shows an analysis that can be visually supported by matrix visualization. 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. The matrix in the middle of Figure 11 depicts exactly the same system, but elements are aligned differently. In this depiction the underlying system structure can be seen clearly: two complete clusters (all elements are mutually linked) and a completely interlinked hierarchy (formed by four elements). The identical structure is shown as a force-directed graph at the right side of Figure 11 making it easy to identify the system structure as well. But even though it is easy to imagine that graph interpretation can become more difficult the more elements and dependencies are included in the structure, so the detection of the hierarchy already requires some experience in interpreting graphs. In comparison, a correctly arranged matrix would show shapes of clusters and hierarchies independent from the amount of implied elements. The example shows that an appropriate matrix alignment can permit the identification of implied structural characteristics. Several authors provide step-by-step instructions that allow manual realignment of matrices (e.g., Kusiak, 1999). With these the visually supported interaction with the structure permits the identification of clusters and hierarchies. However, matrices are less suitable for the representation of multiple structural attributes, if these contain identical elements or dependencies. What is even worse matrix representations can hardly be used for the representation of feedback loops (Lindemann et al., 2008).



	1	2	3	4	5	6	7	8	9	10	11
1						X				X	
2					X			X			X
3				X			X		X		
4			X				X		X		
5		X						X			X
6											
7			X	X					X		
8		X			X						X
9			X	X			X				
10						X					
11	X	X			X	X		X		X	

	4	7	9	3	5	2	8	11	1	10	6
4		X	X	X							
7	X		X	X							
9	X	X		X							
3	X	X	X								
5						X	X	X			
2					X		X	X			
8					X	X		X			
11					X	X	X		X	X	X
1										X	X
10											X
6											

Figure 11. Visually supported interaction with a structure in matrix depiction

The second group of structure analyses uses mathematical computations and applies visualization techniques for mediating the results to the users. Generally, intra-domain matrices are appropriate for representing information about clusters (Hartigan, 1975). But clustering has also been applied in inter-domain matrices (Danilovic & Sigemyr, 2003). Several other analysis results, e.g. start and end nodes, articulation nodes and bridge edges, can be depicted in matrices as well; however graph representations, especially force-directed graphs often provide better understanding for users (Lindemann et al., 2008). An overview of possible analyses and their representation can e.g. be taken from (Gross & Yellen, 2006). Concerning the application of multiple-domain graphs, no implementations can be found in literature that allow intuitive user comprehension – especially if large quantities of elements and dependencies have to be visualized.

Product design application is closely related to the structure analysis discussed before. Often both steps can not be clearly separated in the process of the SCM approach. It is more likely that both steps are mutually linked in intensive iterations (Lindemann et al., 2008). Therefore, the same visualization techniques are appropriate for application in both steps. A problem can appear if users are not used to interacting with networks. In this case they tend to prefer graph representations. Even if matrices are well established in design processes (due to a wide use of spreadsheet software), the reading and interpretation of e.g. the DSM often poses an insuperable barrier to engineers. This explains why DSM-related research is well documented since almost 40 years (Yassine, 2004), but practical applications can be rarely seen (Danilovic & Browning, 2004). In contrast, even everyday problems with complex networks (e.g. railway networks or acquaintance networks) are represented by graph drawings (`http://www.visualcomplexity.com`).

Appropriate model of information visualization

Based on the requirements and possibilities of visual representations a model for the visual support of the SCM approach has been assembled. As it has been shown, one specific system representation can not fulfill the requirements of all five process steps. Therefore, a combination of visualization techniques has been designed and transferred informational content has been specified. The mapping of visualization techniques to the process steps and the identification of required interfaces between them poses the basis for the implementation of a software-based support of the SCM approach.

Figure 12 depicts the model of information visualization for the SCM approach. Three visualization techniques are listed for the initial phase of the system definition. The mind map gets adopted for modeling the system domains and included elements and poses the basis for the subsequent application of the digraph or the intra-domain matrix. Generally, the user can choose between the application of the digraph and the intra-domain matrix, but when there is an interaction with large systems containing a multitude of domains the intra-domain matrix should be preferred.

Information about the linked domains has to be exchanged between the system definition and the step of information acquisition. Now, the already described dependencies between domains have to be detailed into the included system elements with their specific dependencies. For the information acquisition intra- and inter-domain matrices can be applied, as they allow systematic processing of all relevant system dependencies.

Filled matrices, i.e. the information about dependencies between system elements result from the phase of information acquisition. This information gets applied in the step of the deduction of dependencies for computing specific system views. An intra-domain matrix visually supports users in selecting the necessary system parameters for computation (target domain and associated

information sources). This matrix matches the intra-domain matrix applied in the system definition.
Derived intra-domain subsets of the system structure result from the deduction of dependencies. These are passed on to the following structure analysis. During this step, several visualization techniques can be helpful. Their suitability depends on the specific analysis that has to be executed. Therefore, intra- and inter-domain matrices as well as force-directed graphs can be useful instruments for visualization and system interaction by users.
Generally, lists, diagrams and basic digraphs can also be applied for the structure analysis; however, these visualization techniques represent aggregated system views. That means that iterative analyses can become difficult with these visualizations, as analysis results do probably not allow to trace back to the original information about system elements and their dependencies.
The final step of product design application is closely related to the structure analysis. Due to iterative application, both steps can not be clearly separated and apply the same visualization forms consequently.
As Figure 12 shows, several visualization techniques are required for an effective support of the SCM approach. For this reason, available tools that provide single visualizations do not meet all requirements, implying that a combined application of several tools would be accompanied by interface problems. Therefore, a successful support of the SCM approach asks for a tool that implements the visualization model of Figure 12 and allows switching between the visualizations. This requirement goes along with the need for a consistent data model that enables to equally transfer adaptations, which are made in one visualization form to other forms. A consistent data model would also allow to step back to previous phases in the SCM approach. If the analysis results, for example, would not fully meet the expectations, it is possible to add an additional domain in the step of system definition – and pass through the procedure again.

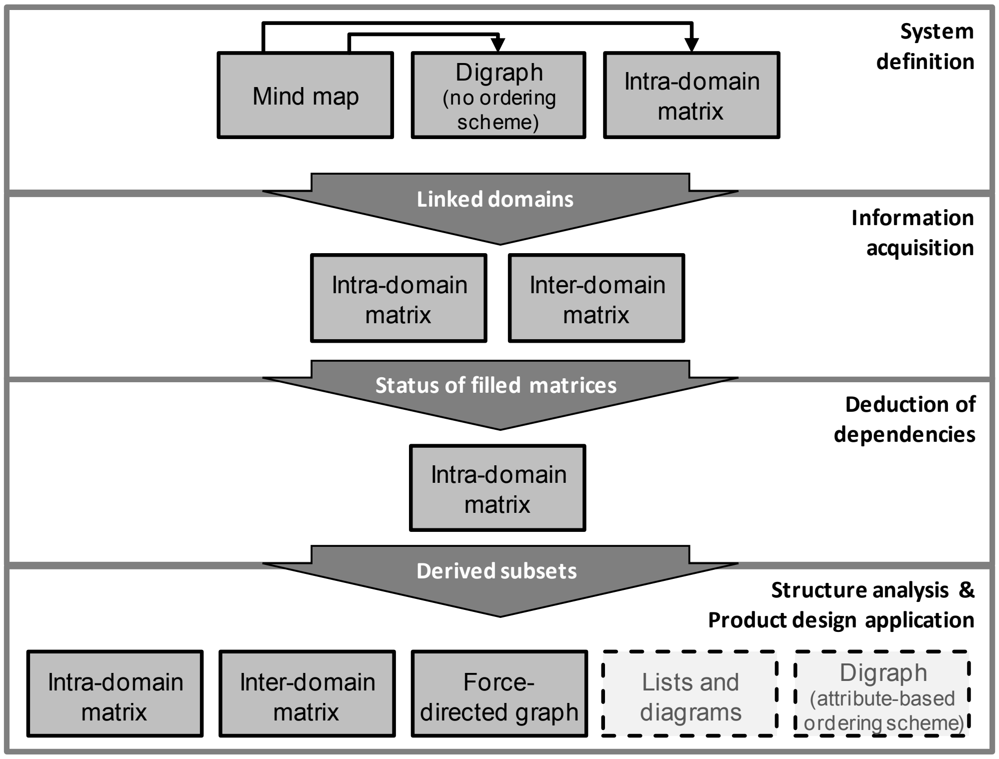


Figure 12. Information visualization for the Structural Complexity Management approach

Conclusion and outlook

Many visualization approaches are available for the purpose of managing complex networks. They are all suitable for the application on specific problem situations but do not support an entire process of problem solving. The Structural Complexity Management (SCM) approach represents a comprehensive approach for the handling of multiple-domain systems containing complex dependency networks. Whereas the approach describes the whole procedure from initial system definition until the final application of improvements of product design, an accompanying description of visualization techniques and their useful combination has not been provided yet. For this reason, the practical application of the SCM approach has been limited to the handling of small systems, so far. Several tools apply information visualization that is qualified for supporting specific phases and tasks of the procedure. But information exchange between those tools is difficult to implement and complicates practical use.

Due to these limits in practical application we identified suitable visualization techniques and matched them with the requirements of the SCM approach. Based on this procedure we created a model of combined visualization techniques and described the required information exchange between them. The ongoing implementation of this visualization model will help to increase the applicability of the SCM approach to industrial use cases of complexity management significantly. The software tool LOOMEO (`http://www.teseon.com`) is used as a basis for the implementation of the elaborated information visualization model. Until today, we have established the combined application of intra-domain matrices and force-directed graphs (Figure 13). This implementation already allows us to support large parts of the SCM approach. In the future, the visualization support will be consequently enhanced until the entire process can be supported effectively.

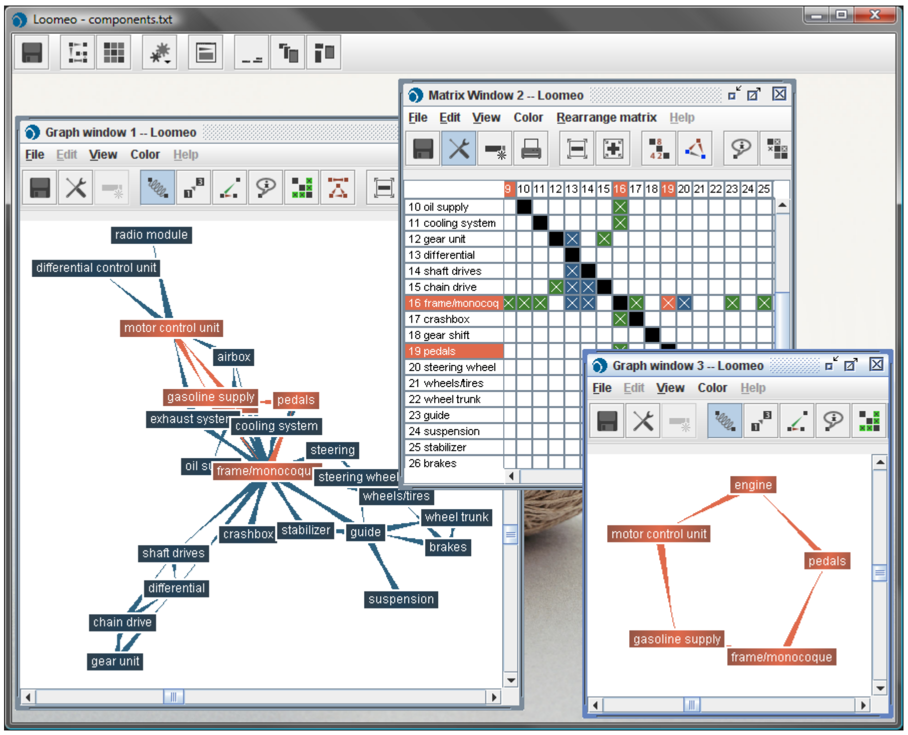


Figure 13. Simultaneous interaction in inter-domain matrices and force-directed graphs

References

Akao, Y. 1992. *QFD - Quality Function Deployment.* Landsberg a. L.: Moderne Industrie 1992.

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

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.

Bollobás, B. 1990. *Graph Theory*. New York: Springer.

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: ASME.

Browning, T. 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. In *IEEE Transactions on Engineering Management 48 (2001)* 3, pp 292-306.

Buzan, T. 1996. *The Mind Map Book*. Penguin Books.

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

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.

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.

Gross, J. L., Yellen, J. 2006. *Graph Theory and its Applications*. 2nd Ed., Boca Raton: Chapman & Hall/CRC.

Hartigan, J. A. 1975. *Clustering Algorithms*. New York: John Wiley & Sons.

A. Kusiak. 1999. *Engineering Design – Products, Processes and Systems*. San Diego: Academic Press.

Lindemann, U., Maurer, M., Braun, T. 2008. *Structural Complexity Management – An Approach for the Field of Product Design*. Berlin: Springer.

Melnikov, O., Tyshkevich, R., Yemelichev, V., Sarvanov, V. 1994. *Lectures on Graph Theory*. Mannheim: BI Wissenschaftsverlag.

TESEON 2008. `http://www.teseon.com`.

Visual Complexity 2008. `http://www.visualcomplexity.com`.

Ware, C. 2004. *Information Visualization – Perception for Design*. 2nd Ed. Amsterdam: Elsevier.

Yassine, A. 2004. An Introduction to Modeling and Analyzing Complex Product Development Processes Using the Design Structure Matrix (DSM) Method. *Italian Management Review 9*

BIOGRAPHY

Dr.-Ing. Maik Maurer graduated in mechanical engineering at the TU München, Germany and earned his doctorate in engineering with a dissertation on structural aspects in product design. In 2006 he co-founded the company Teseon that provides solutions for complexity management in industrial applications.

Dr.-Ing Thomas Braun graduated in mechanical engineering at the TU München, Germany and earned his doctorate in engineering with a dissertation on methods and tools for the strategic product planning. In 2006 he co-founded the company Teseon that provides solutions for complexity management in industrial applications.

Prof. Dr.-Ing. Udo Lindemann was postdoctoral engaged in many industrial enterprises. Since 1995 he is head of the Department for product development of the Technical University of Munich. Teaching and research are focused on the development of strategies for the early stages of development, approaches to product innovation, questions of cost management and the use of computers in product development and the inclusion of psychological and sociological insights.