Competence in Design and Development
CiDaD Working Paper Series
ISSN 1861-079X – Januar 2007 – Jahrgang 03 – Nr. 01

Facing Multi-Domain Complexity in Product Development

Maik Maurer, Udo Lindemann

The Impact of Complexity

Complexity in product development is continuously increasing, and yet it has not been satisfactorily addressed in literature and practice. The quality problems and recalls in today's car industry give a striking example. Product complexity is driven by the market demand for individualized and multifunctional products (Pine 1993). The market itself has become more global and specialized (Anderson 2006). To fulfill individual customer demands or local regulations and specifications, product and variant diversity have to increase (Ericsson and Erixon 1999). Consequently, controlling and handling complexity in product development processes has turned into an important issue, as process diversity increases with the quantity of product variants and process steps become ever more intensely interconnected. Finally, organizational structures of companies are affected by the increased complexity (see figure 1). Market demands, product diversity and flexible business processes require new concepts and strategies in organizational design to meet increasing interdependencies between people acting in the development process (Lindemann et al. 2006). Product adaptations, as they are required by product individualization or mass customization (Piller 1998), affect all aspects of product generation and require appropriate methods of complexity management.

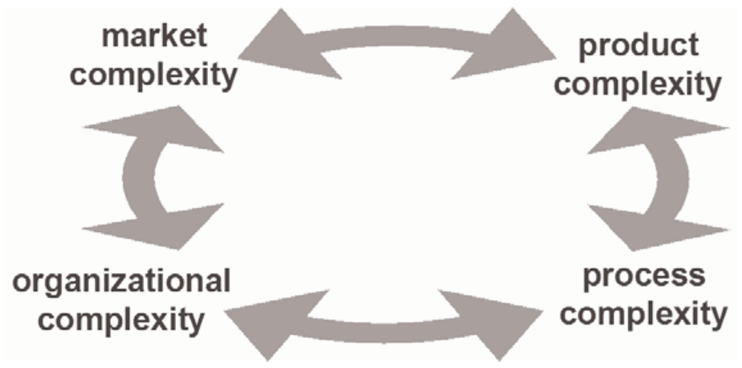

Fig. 1 Aspects of complexity

Inhalt

Lehrstuhl für Produktentwicklung
Prof. Dr.-Ing. Udo Lindemann
Boltzmannstr. 15
D-85748 Garching b. München
Tel. 089/289 15131
Fax 089/289 15144
Internet: www.pe.mw.tum.de

PRODUKTENTWICKLUNG *product development*
TECHNISCHE UNIVERSITÄT MÜNCHEN

Most approaches for controlling product complexity focus primarily on its reduction (Schuh 2001; Blackenfelt 2001; Riitahuhta & Pulkkinen 2001). Although it is serves a purpose to avoid unnecessary complexity, it may not to be advantageous to reduce complexity at any cost. Complexity often relates directly to attributes relevant to the customer; thus complexity reduction may decrease competitiveness (Lindemann et al. 2005). If a competitive product permits customization, it will appeal to more customer groups than a standardized product.

A positive aspect of controlled product complexity is its use as a barrier to product plagiarism (Wildemann et al. 2007). 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.

The understanding of the product structure (inter-connectivity of components) is essential when interacting with a complex system. A multitude of system features are only implemented by their structure rather than by their components (Lindemann et al. 2005). For example, system robustness is a feature that consists of a comprehensive combination of system elements. The demand for robust systems is directly related to the control of its complex interdependency network.

Modern product design is characterized by shortened development cycles. Newly arising requirements are mostly realized by adapting existing products. The changes resulting from such adaptations may have an unexpected impact on a number of interrelated components. If such impact is not considered at the beginning of the development process, delays due to required iteration loops are inevitable. An effective system for controlling complexity permits the prediction of change impact that previously would have gone unnoticed (Clarkson et al. 2001). Often, such impact concerns different development domains, company departments, and positions of responsibility (Eichinger et al. 2005). The practical benefit is derived from a sound basis for making decisions on the feasibility and necessity of planned changes.

One major requirement for defining such robust systems is to identify significant structural characteristics and then to derive suitable measures. This paper introduces a comprehensive methodology for analyzing, controlling and optimizing complex systems containing a multitude of interdependencies between several domains such as the element types components, functions, documents etc.

Methodology

The presented methodology consists of three steps: definition of the system, multi-domain-analysis with derivation of relevant subsets, and analysis of selected network constellations. The system definition covers collecting case specific available input data and desired output data as well as linking the two. Output data are generated during multi-domain analysis are subjected to detailed analysis and interpretation in the third step. Results provide for a better understanding of the system and serve as a starting point for targeted system optimization.

Specific visualization forms, design structure matrices and strength-based graphs are used to represent network information to the user. As design structure matrices (Browning 2001) typically focus on interdependencies between elements of one domain (e.g., interdependencies between product components), the approach is extended to the interaction between multiple domains (Eichinger et al. 2005). Although the combination of different domains (e.g. components, functions, and documents) enables formulating important system information, representations for users are confined to interdependencies within the same domain (according to the design structure matrix approach). Thus a large number of available algorithms can be applied and visualization becomes intuitive for the user.

System definition

The objective of this approach is to derive relevant relationships of elements within one domain. A case specific dependency type must be implemented for relations between the elements. The resulting networks, which can for instance depict interaction between product elements, can be analyzed by a multitude of available algorithms and are suitable for communicating structural content to users.

Modeling approaches usually contain numerous different elements and relation types simultaneously (e.g. process plans) and contain highly complex information. However, these networks are difficult to gain access to using analysis algorithms and the structure visualization is too complex to permit intuitive comprehension. Models created with this approach must be chosen carefully regarding the case specific objective, because analysis and optimization by algorithms is limited to one element type.

The first task in system definition is to identify the domains which have to be considered for the problem. These domains can be selected from a list of standard domains forming most of the systems considered in general (Jarratt 2004; Pimmler and Eppinger 1994). Examples of standard domains are requirements, functions, components, data, documents, roles, process steps, or milestones. Although these domains will fit for a multitude of situations, the applied level of detail must be chosen carefully. If domain elements are modeled at too general a level, results will not permit further system insight. However, too detailed modeling of domain elements will result in enormous amounts of data, complicating analysis and interpretation. As the suit-

able level of detail can typically not be determined in advance, it is often helpful to model domain elements hierarchically. Later on, the appropriate detail level can be selected and known interdependencies between elements can be transferred to this level.
Once the concerned domains are identified, interdependency types between them have to be determined. For example, the domain “components” can be linked to “function” by the dependency meaning “realizes”. Generally, one must differentiate between the interdependency meaning of existing data (that will normally be given) and the meanings required for solving the problem.
It is recommended to arrange the domains in a symmetric matrix as is shown in the example in figure 2. Networks consisting of only one domain type (according to the approach of the design structure matrix) are located on the matrix diagonal. According to the systematic combination of domains in a square matrix, this matrix can be called multi-domain matrix (MDM) or multi-domain design structure matrix (MDSM).

	Components	People	Data	Processes	Milestones
Components	Component impacts component	Person works on component	Component represented by data		Component completed at milestone
People			Person generates data		
Data	Data required for component	Data required by person		Data required for process	Data available at milestone
Processes		Person works on process	Process generates data	Process information transfer	Process completed at milestone
Milestones					

Fig. 2 Example of a multi-domain matrix with link meaning.

The dependency meaning between two domains is shown in the matrix cells. The matrix supports a systematic procedure where every possible domain linking is addressed. It may occur that more than one dependency meaning is appropriate for connecting two specific domains. For further analysis it is important that these dependency meanings are stored in separate matrices, because merging will prohibit the application of most algorithms.
First, one must define the meaning that can be extracted from existing data (e.g. from databases or interviews). Important criteria for the selection are data availability as well as reliability. This selection is case specific and must be done carefully, as further steps are based on this data. Regard-ing the linking of components in figure 2, a relation between two components could represent a spatial link or a general change impact. Interdependency meaning in one domain linking network must be identical for all interdependencies in order to allow later analysis and interpretation. If different linkage types between the same two domains are possible, related data must not be mixed.
As shown in figure 2, two matrices exist that link the same two domains (e.g. components and people), differing in inverted link order. If one dependency meaning between people and components represents “person works on component”, the second dependency meaning may express “component is processed by person”. Thus, the second meaning corresponds to the first one as its passive counterpart. The acquisition of both dependency meanings can be useful if information is extracted from different sources. In this case, further analysis permits validation by comparing this information. If both mutually related matrices do not emerge from different sources, it is not useful to fill in both, because the opposite matrix can simply be transposed.
After specifying available data input, target domains have to be selected, which are visualized and serve for analysis and interpretation. In addition, it must be specified how these target domains can be determined using available domain meanings. This task can be done by matrix multiplication. One intradomain network (corresponding to DSM, e.g. linking of components to components) can be determined by different data input. It is useful to consider such networks simultaneously, as supplementary information can be obtained.

Multi-domain analysis

The next methodical step is to compute intra-domain networks relevant to the problem. Depending on case specific structural characteristics (e.g. sub graphs such as feedback loops or hierarchies), derived intra-domain networks may or may not provide useful information. For this reason it makes sense to set up a computation routine that permits intra-domain networks to be determined within a short space of time and hence permits a larger number of networks to be generated from the given domain meaning. Significant networks can then be selected for further consideration.
Generally, there are four possibilities for computing an intra-domain network from dependencies between two different domains (matrices connecting elements of two different domains are named inter-domain matrices). All possibilities are explained in the following using the same example:
Dependencies between people are deduced by dependencies of people to documents. This could represent a typical scenario in product development, where designers are dependent on other designers due to data transfer.
Of course the intradomain network of people can be addressed directly without regarding a second domain. This could represent the existing team structure. However, it is often easier to take a detour for the following reason: to acquire dependencies between people directly, people must be asked to whom they are connected; this must be answered under the condition that a document transfer takes place between them (Sosa et al. 2004). It is obvious that this question would be hard to answer resulting in a network of poor quality.
However, if documents are introduced as a second domain, the question is answered more easily: now people must declare what documents they provide and which ones they require. The required dependencies between people can then be deduced from information obtained.

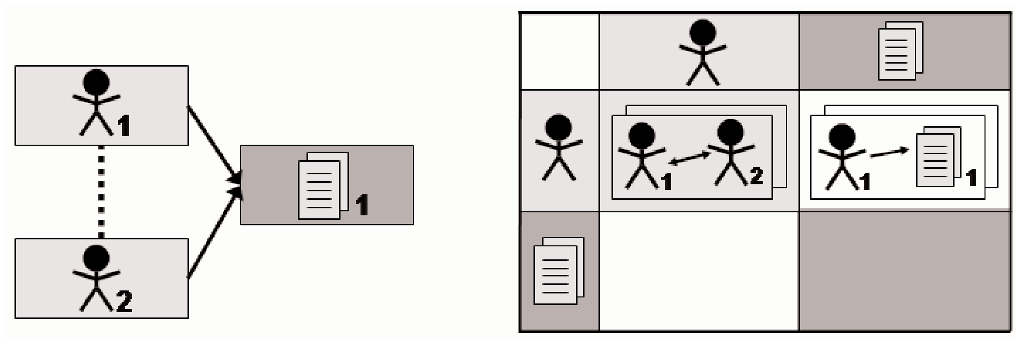

Fig. 3 Computing an intra-domain network from one inter-domain network.

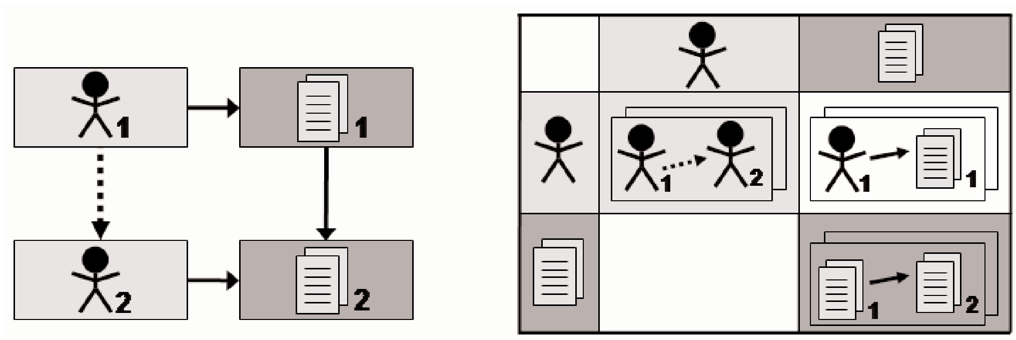

Fig. 4 Computing an intra-domain network from one inter-domain and one intra-domain network.

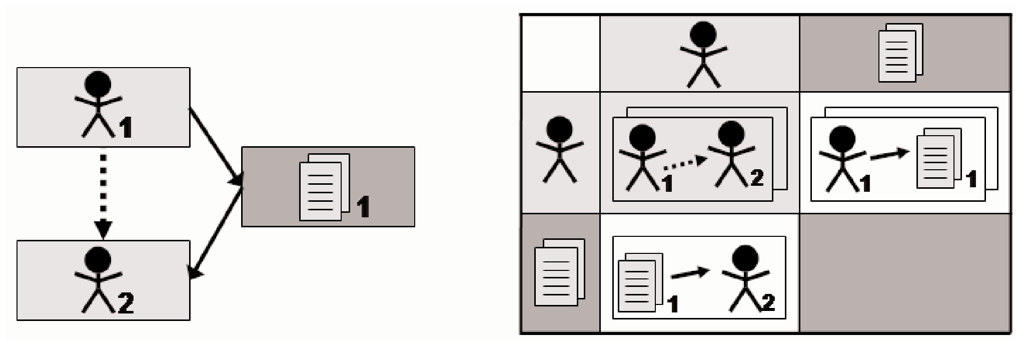

Fig. 5 Computing an intra-domain network from two inter-domain networks.

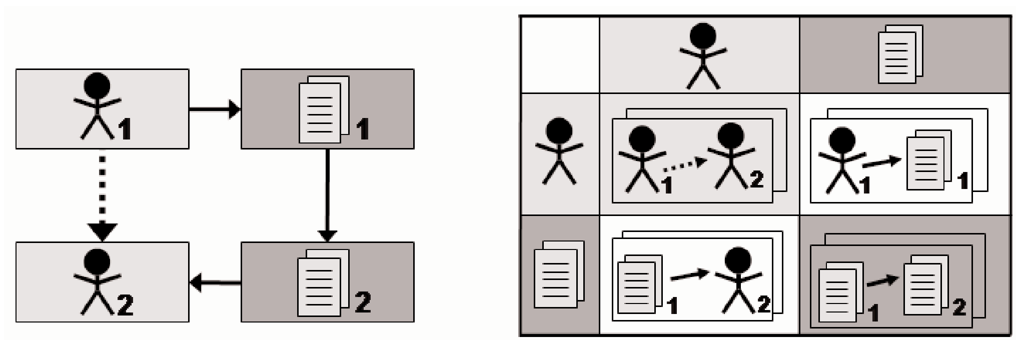

Fig. 6 Computing an intra-domain network from one intra-domain and two inter-domain networks.

- Case 1: An intra-domain network is derived from one inter-domain matrix; figure 3 shows the logic dependencies as well as the matrix alignment in the context of the chosen example, i.e. dependencies between people are deduced from people's links to documents. The generated intra-domain network does not provide a linking direction; two people are linked because they work on or access the same document.
- Case 2: An intra-domain network can be derived from one inter-domain and one intra-domain matrix; in this case (shown in figure 4), person 1 and person 2 both work on separate documents; however document 1 is required as input for working on document 2, which causes person 1 to be dependent on person 2.
- Case 3: An intra-domain network can be derived from two inter-domain matrices; the example in figure 5 shows that a dependency directed from person 1 to person 2 can be computed because person 1 works on document 1 which is required by person 2.
- Case 4: An intra-domain network is derived from two inter-domain and one intra-domain linking; figure 6 shows that a dependency is directed from person 1 to person 2, because person 1 works on document 1, which is required for the compilation of document 2; finally, document 2 is required by person 2.

It is hardly possible to define precise criteria for pre-selection regarding the computed intra-domain networks. However, the general idea of the presented approach is to identify and analyze relevant structures that have so far have gone unnoticed. Therefore it is difficult to reliably assign suitable networks upfront. The following criteria are useful to narrow down the number of relevant networks in most cases:

If all network elements are mutually interconnected without any distinctive difference in relation weights, networks will not provide helpful information, because they do not inherit any structural significance. Networks representing the same domain as those created by original (non-computed) data may be useful for closer analysis if they are significantly different from known networks. For example, it can provide further system understanding, if interdependencies between people are gathered by interviews and are compared to a people network computed by people's assignment to components or data. Differences in these networks will often be the starting point for optimization measures.

It is an important aspect to take the quality of ascertainable information into account, as significance of computed networks directly depends on this input information.

In addition, some networks will be derivable from existing systems (databases, PDM-systems, bill of material etc.). This influences the selection of considered input data as well as the effort of information acquisition is a lot lower and quality of input data is higher than with interviews.

Analysis of selected intra-domain networks

A large set of analysis criteria is available for considering intra-domain networks that were computed by the presented multi-domain approach more closely. All analyses are based on the possibilities of graph theory, but may require different representation forms (matrix, graph, list, diagram etc.). Table 1 compiles relevant basic criteria for analysis and specifies a possible interpretation. The shown example refers to an intra-domain network of components. The interdependency meaning describes change impact between components. However, interpretations are rather case specific and may depend on structural conditions, the origin of considered data, and the meaning of the network in question. Thus transferring interpretations between different use cases can be problematic even if criteria are similar. Basic analysis criteria have to be considered as they appear together so it is almost impossible to provide rules of thumb; believing that isolated values provide the whole picture may often be misleading.

Some of the analyses are suitable to be applied to the initial network in order to provide a general overview. Others are to be applied in combination or only to selected parts of the network. For this reason, software support is absolutely indispensable and must permit flexible use of these analyses.

Most analysis criteria shown in table 1 are already applied in specific methods of product development. E.g. activity and criticality represent element characteristics in influence portfolios (Daenzer 1999). Generally, those applying analysis criteria in development methods have been rather selective so far; by contrast, the approach presented here provides the implementation of a comprehensive set of criteria in order to improve applied analyses.

Table 1 Analysis criteria for intra-domain networks.

Analysis criterion	Explanation	Example
Active sum	Quantity of outgoing relations	Components of highest impact
Activity	Division of active sum by passive sum	Degree of active participation
Articulation node	The only node connecting two sub graphs	Change impact must pass by this node
Bi-connected component	Sub graph only connected by articulation node or bridge edge	Canceling one relation prevents propagation of probable change impact
Bridge edge	The only edge connecting two sub graphs	Change impact must pass by this edge
Criticality	Multiplication of active sum and passive sum	Degree of network integration
Distance	Specifies the distances between nodes in a structure	High values represent indirect impact to/from node
End node	Node possessing only incoming (passive) relations	Component only receives change impact
Feedback loop	Circular sub graph	Change impact eventually affects the originating node
Hierarchy	Node branching out in different levels	Spreading change impact originates from a specific element
Isolation	Nodes without any relation to other parts of a structure	Component is not affected by/ does not cause change impact
Locality	All nodes directly connected to a central node	Nodes connected by outgoing edges receive direct change impact
Passive sum	Quantity of incoming relations	Amount of change impact a component receives
Proximity	Specifies the distance from other nodes in the graph	Node with high proximity receives/provides direct impact
Reachability	Node can be reached from other nodes by dependency paths	Node with high reachability receives/provides impact from many nodes
Shortest path	Shortest connection between two nodes	Most probable change impact propagation
Spanning tree	Sub graph connecting all system nodes	Required relations for reaching all system elements
Start node	Node possessing only outgoing (active) relations	Component only spreads change impact
Strongly connected part	All nodes can mutually be reached by a specific path	Each node can possess change impact to any other node in the sub graph
Triangularization/ Sequencing	Defined as sequential or block order of nodes	Best sequential order for engineering change

Case study

The case study observes the student group "TUfast" located at the Faculty of Mechanical Engineering at the Technical University of Munich. The team develops a racecar compliant with the Student Formula regulations each year. For the competition, the car design is evaluated from different perspectives: handling, performance, engineering and design to cost.

The management of complex dependencies plays a major role 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 finetuned to be equally successful in all the different 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 respects, similar to the automotive industry. The design process is highly distributed, involving about 35 engineering students in a concurrent design environment. Therefore typical problems such as the management of releases, the meaningful modularization of both product and the development process and an optimized distribution of information are apparent.

The TUfast group relies on very rapid development processes (i.e. time constraints) because a new racecar has to be developed once a year. Although experience from previously completed 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.

The following five domains were identified for the layout of the scenario: components, people, data, process steps, and milestones. The dependencies were directly recorded from the group by interviews and shaded grey in figure 7. The relevant dependency meaning is given in figure 2. Altogether 14 different matrices were available as original data; eleven are inter-domain and three are intra-domain matrices. Based on this data five intra-domain networks were analyzed in detail. Black arrows show how matrices are determined in figure 7. Component and process intra-domain networks already existed as original data (round arrows); the other five networks are each computed from two related networks according to the straight arrows. Finally, some available matrices were not used at all due to the low quality of the input data (e.g. data-components).

Matrix multiplication permits the deduction of 38 intra-domain networks from the available data. However, this quantity can already be reduced to 16 relevant structures because 22 networks are computed from one single inter-domain network (which does not provide any significant system insight).

For this use case a rather general system level was applied. System domains contain between ten and 30 elements each, corresponding to a reasonable effort to compute all 16 networks for carrying out pre-analyses. Finally, seven intra-domain networks were selected for detailed analysis.

At first, the detailed analysis of the component network representing change impact between components is shown. Figure 8 displays an overview in a strength-based graph. The advantage of this method is an alignment which is intuitive for users. Graph nodes repel each other and are mutually attracted by graph edges. It is possible to assign weights to graph edges. Edge weights result in different attraction forces (leading to different element distances) between nodes. It can easily be seen that the component structure is composed of three major blocks meeting in the central element frame/monocoque.

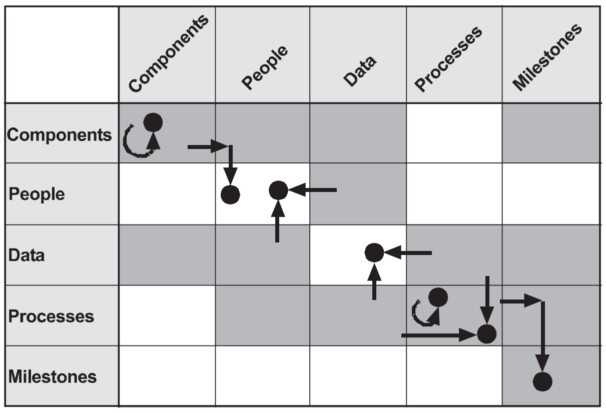

Fig. 7 Determination of intra-domain networks from available data.

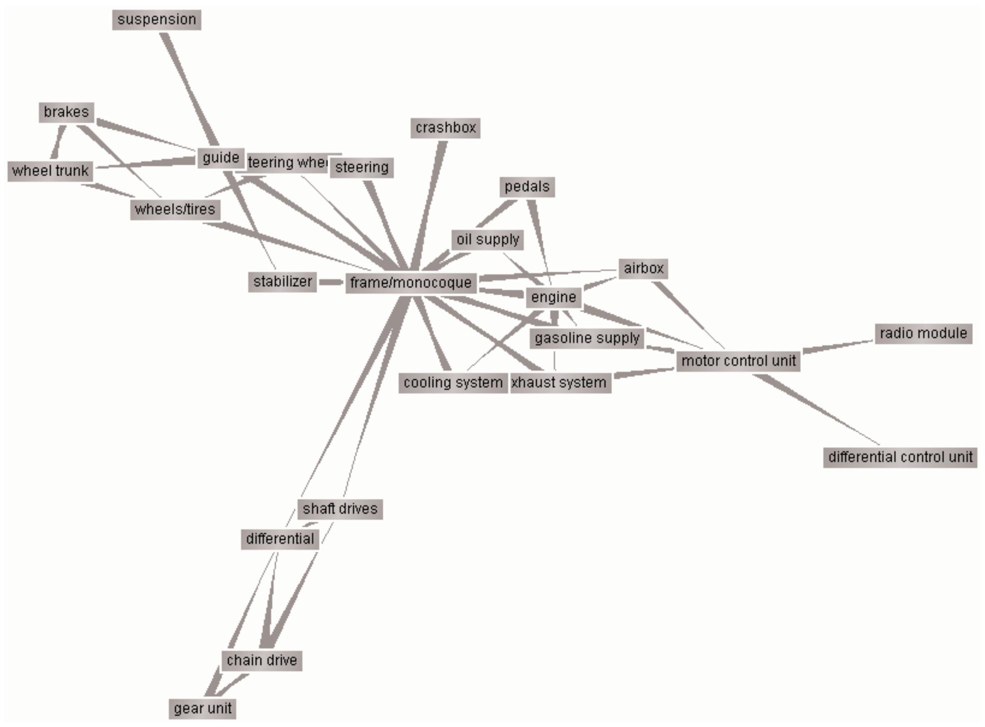

Fig. 8 Graph visualization of the component intra-domain linking.

There are two strongly connected parts in the network containing 18 and 2 nodes. Three elements (data logging, cable loom, and gearshift) are isolated from the rest of the structure (not shown in figure 8) and therefore do not provide or receive change impact. Differential, differential control unit, and radio module are end nodes which only receive change impact from other components. Besides the frame/monocoque, the guide and the motor control unit are the articulation nodes representing the only connection between two sub graphs. The brakes mostly receive indirect change impact and the airbox mostly spreads indirect change impact. The following interpretation can be derived from the component network:

- The identified blocks must be coordinated when elaborating design details
- Frame/monocoque is to be included in each bidirectional coordination
- Changes in one strongly connected part can be conducted without considering the other one
- Components represented by isolated nodes can be adapted independently from all other components.

Some recommendations can be made in order to improve the component based design process: The need for coordinating component adaptations can be reduced if people are assigned to components located in the same block. Team organization gets easier as the block's internal is greater than its external need for coordination. Consequently, small blocks consisting of, say, only two components should be assigned to only one person. Obviously, component adaptation and resulting change impact cannot be avoided. However, components spreading enormous impact to others (e.g. frame/monocoque and engine) 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 (sub graphs extracted from the graph representation). There are possibilities for reducing the quantity of required component change if the radio module and differential control unit (end noted in the graph) as well as the airbox (mostly indirect impact to other components) are defined late in the design process. Otherwise the brakes (start node in the graph) should be specified early in the design process, because impact by other components is not to be expected.

The network of change impact between components was acquired by interviews. People will often know about specific structural characteristics even if far reaching or overlapping impact will hardly be explicitly expressed. However, the inter-domain network of people was not available explicitly and has therefore been deduced from component and data views.

At first, the network between people has been computed by the intra-domain dependencies between people and components. The interdependency meaning of the resulting network is: two people are interrelated, because they work on (different) components that have a change impact on each other. The dependency between two people is directed according to the direction of change impact between associated components. Different weights of dependencies may result if two people are linked by more than one component change impact. Figure 9 illustrates part of the resulting intra-domain network in a strength-based graph; this representation focuses only on bidirectional dependencies between people. People located close to each other work on several components which are interdependent by 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 importance of cooperation, because the closer two people are,

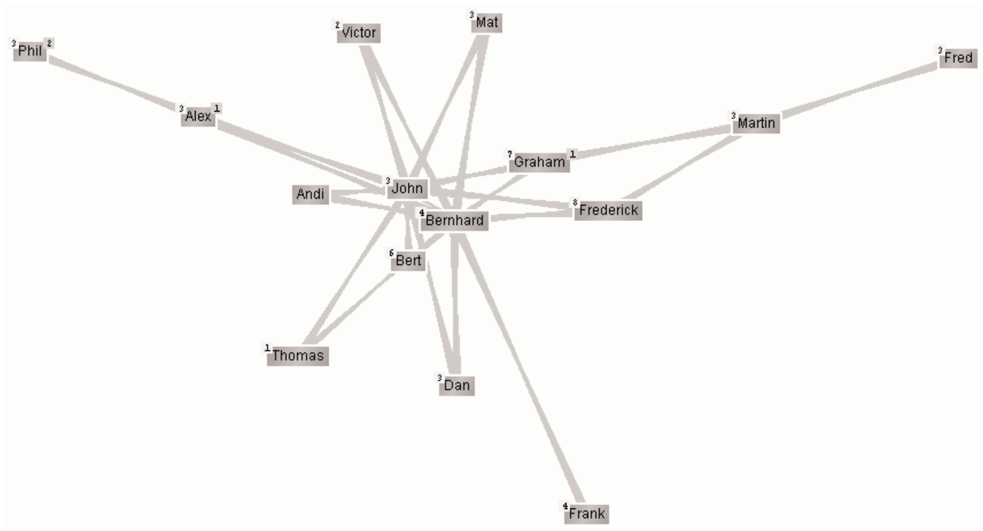

Fig. 9 Sub graph of linking between people based on people's links to components.

the more change impact affects them.

Nine complete clusters exist comprised of three people each. The same two people, Bernhard and John, are involved in all of these clusters. 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.

So far all team members take part in the weekly team meeting, but figure 9 suggests reconsidering people's coordination. Many team members require bidirectional coordination that can not perfectly be executed in the general meeting (or else it becomes too time consuming for the other members). Furthermore, a group of three people form the core of the design team with an enormous demand for internal as well as external coordination. The decision to arrange an overall team meeting probably arose from a lack of knowledge of the coordination needs. It would be useful to prioritize bidirectional exchange based on the quantity of component change impact. People should ensure they possess adequate means of communication, when they are located at a network position that requires intense coordination.

Figure 10 shows the structural relevance of hierarchical subsets. The illustrated sub graph represents one of 15 hierarchies included in the entire network of people (based on peoples' assignment to components and components' change impact). A hierarchy in this network expresses how the communication needs propagate due to component adaptation executed by one person. In figure 10, Phil represents the top element (initial point) of the hierarchy. If one would only consider persons within his direct surrounding (nodes connected by dependencies), the situation might be underestimated, as only three people are directly affected due to component change impact. However, change impact caused by Phil may spread to up to 17 other people if subsequent dependencies are taken into account. A useful method for practical application would be to filter the paths of change propagation that may cause highest change impact. In the example in figure 10, these would be the linking paths from Phil to Bernhard and John (passing by Graham and Alex). 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 required resources). Such checklists must be deduced individually for every person who at the initial point of an identified hierarchy.

A further important dependency network of people arises from related data. An intra-domain network of people has been computed by two complementary inter-domain networks which connect people and data. Hereby, the interdependency meaning is "person generates data" and "data required by person". The resulting dependency meaning is: "an edge is directed from one person to another if the first one generates data the second one requires". The arising network of people is displayed in figure 11. Although it seems to be rather complicated (due to the quantity of dependencies) the structure is highly significant. Generally, two different groups of people can be identified within the given 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 data as well. The second group of people is characterized by their strictly passive dependencies based on data exchange. These people require data as input specification for their design tasks but the work results are not fed back into data descriptions any more. There are only few people who do not belong to one of these two groups.

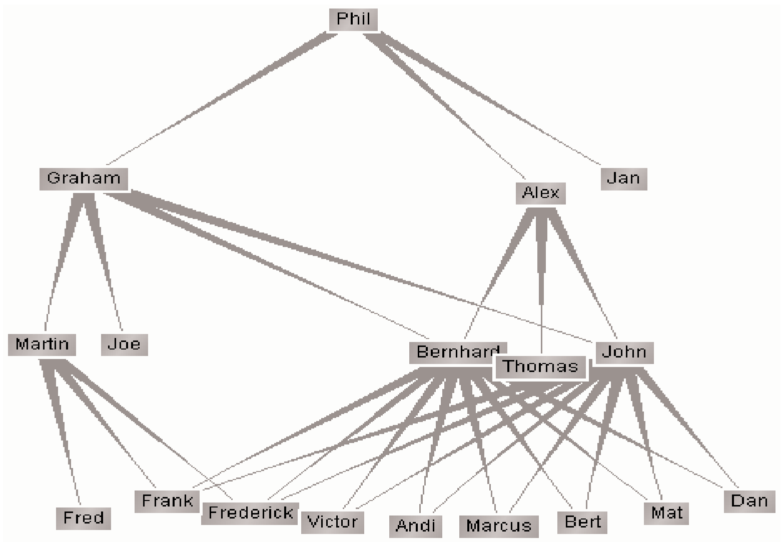

Fig. 10 Hierarchical sub graph filtered from linking of people.

Possible optimization measures concerning the design process are: on the one hand, people who generate data should know who receives their output. This could be realized by visualizing the data recipients immediately surrounding each individual. This facilitates the push principle of information flow. On the other hand, people depending on data provision should know about their suppliers. Visual support of individual supply chains can permit a pull principle, for example in the case of supply delay. Furthermore, the network of people based on data exchange allows for the planning of fluctuation in the team members: 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 tasks but the information flow will not be disturbed because other people do not depend on information provision by the team member in question. The situation gets 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 attend the person's tasks; in fact, 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. For these reasons, it is recommended to set up a structured plan for the replacement of such team members.

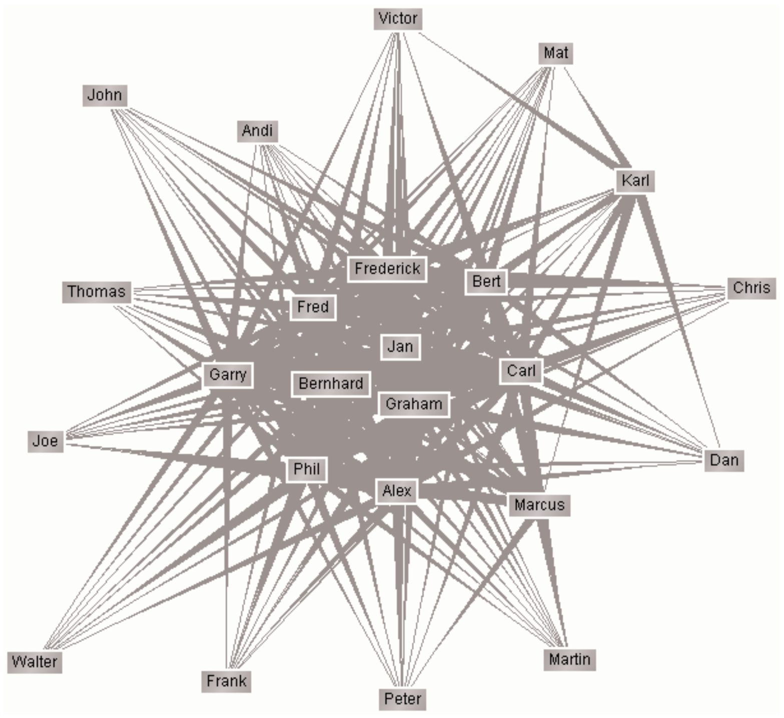

Fig. 10 Hierarchical sub graph filtered from linking of people.

Conclusion

The presented approach provides methodical support for optimizing complex design processes through consideration of the entire range of involved domains. The use of multi-domain matrices allows for improved data acquisition, because complex dependency types can be separated into simpler questions more suitable for interviews. Furthermore, the approach of multi-domain matrices makes it possible to systematically derive intra-domain networks which are suitable for algorithmic analysis as well as intuitive user comprehension. Analysis of intra-domain networks can be executed by means of structure criteria known from graph theory; a list of relevant criteria for development processes has been given.

When applying the multi-domain approach it is important to ensure the availability of input data. Extraction of data from existing data bases is much less time consuming and promises higher data quality than acquisition by interviews. However, data collected in interviews may offer interesting insights, as it may contain information that has never been collected systematically. Such data often stems from the designer's experience.

Various methods that promote general system understanding are used in practice. In comparison to these, the multi-domain matrix is characterized by the ability to integrate several aspects of development processes simultaneously. Once the multi-domain matrix is on hand, networks answering specific questions can be deducted automatically with at little additional expense. However, initial system definition and data acquisition may be time consuming and experience is needed to guarantee good data quality.

The case study illustrated the possibilities of the approach but some facts seem to be problematic: so far no rules are available to reliably arrive at detailed network meaning from analysis criteria. For this reason, comprehensive experience in system modeling and analysis is required. If people rely on the results of a single analysis or only a few system characteristics, misinterpretation is likely to occur.

ACKNOWLEDGMENTS

The authors would like to thank the team TUfast for the excellent collaboration and provision of the case study data. Furthermore, we thank the PLEGMATUM group for ideas and helpful discussions on the topic.

References

Anderson C: The Long Tail - How Endless Choice is Creating Unlimited Demand. London: Random House Business Books 2006

Blackenfelt, M.: Managing complexity by product modularisation. Stockholm: KTH 2001

Browning T: "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions", IEEE Transactions on Engineering Management, 48(3), 2001, pp 292-306

Clarkson P J, Simmons C, Eckert C M: Predicting Change Propagation in Complex Design. In: Proceedings of the DETC 2001 ASME 2001 International Design Engineering Technical Conferences. 13th International Conference on Design Theory and Methodology, September 9-12, 2001, Pittsburgh, USA 2001

Daenzer W F: Systems Engineering. Zürich: Verl. Industrielle Organisation 1999

Eichinger M, Maurer M, Pulm U, Lindemann U: 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, Italy, July 4-7 2006

Ericsson A; Erixon G: Controlling Design Variants – Modular Product Platforms. ASME Press: New York 1999.

Jarrat, T, A Model-based Approach to Support the Management of Engineering Change, Ph.D. the sis, University of Cambridge, Cambridge, UK 2004

Lindemann U; Maurer M; Kreimeyer M: Intelligent strate gies for structuring products. In: Clarkson J; Huhtala M (Eds.): Engineering Design - Theory and practice. Cambridge, UK: Engineering Design Centre 2005

Lindemann U, Reichwald R, Zäh M F: Individualisierte Produkte - Komplexität beherrschen in Entwicklung und Produktion. Berlin: Springer 2006

Piller F T: Kundenindividuelle Massenproduktion. München u. a.: Hanser 1998

Pimmler T U, Eppinger S D: Integration analysis of product decompositions, In: Proceedings of ASME Design Theory and Methodology, Minneapolis, 1994, pp 343-351

Pine J: Mass Customization – The New Frontier in Business Competition. Boston: Harvard business School Press 1993

Riitahuhta, A; Pulkkinen, A: Design for configuration. Berlin: Springer 2001

Schuh G: Produktkomplexität managen - Strategien, Methoden, Tools. München: Hanser 2001

Sosa M E, Eppinger S D, Rowles C M: The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. In: Management Science, Vol. 50, No. 12, December 2004, pp 1674-1689

Wildemann H, Ann C, Broy M, Günthner W A, Lindemann U: Plagiatschutz - Handlungsspielräume der produzierenden Industrie gegen Produktpiraterie. München: TCW Transfer-Centrum GmbH & Co. KG 2007

Further information

Maik Maurer

Lehrstuhl für Produktentwicklung
Boltzmannstr. 15
85748 Garching

Telefon: 089 28915155
Telefax: 089 28915144
Email: maik.maurer@pe.mw.tum.de

Impressum

Die CiDaD-Working Paper Series werden herausgegeben von:

Lehrstuhl für Produktentwicklung
Technische Universität München
Boltzmannstrasse 15
D-85748 Garching (Germany)
Tel. +49 (0)89 289 15131
Fax +49 (0)89 289 15144
http://www.pe.mw.tum.de

Verantw. i.S.d.P.
Prof. Dr.-Ing. Udo Lindemann
udo.lindemann@pe.mw.tum.de

Redaktion
Dipl.-Ing. Maik Maurer
maik.maurer@pe.mw.tum.de

Gestaltung, Grafik- und Bildbearbeitung
Rainer Hinterberger
rainer.hinterberger@pe.mw.tum.de