A Complexity Management Framework

6

The previous chapters of this thesis provide an overview of engineering practice, historic development and the current scientific classification of complexity management approaches. But when one is confronted with a specific complexity challenge, which approach should one apply? The description of the historic development especially indicates a trend towards increasing system complexity—and with it more sophisticated models for its management. But does this mean that a classic reductionistic system model is no longer suitable for solving complex challenges? It depends, of course, on the situation.

A systematic approach towards complexity management is crucial, because complex challenges are typically characterized by a lack of clarity. But such an unclear initial situation does not allow for well-founded selection of measures to overcome complexity. In addition, complex challenges often come with high urgency for action, which can mislead people to take quick measures without assuring their suitability to the situation.

Successful complexity management requires identification of the underlying causes. Otherwise, there is a high risk of treating only the consequences of complexity, which then does not lead to a sustainable solution. It is important to clearly define the system under consideration, which means determining the system's boundaries, included elements and interdependencies between them. Such a formal definition helps to identify the type of complexity at hand, if it is about internal or external complexity and if it originates from the market, product, process or organizational area. The system definition also enables the determination of whether the considered complexity is useful or useless. Based on all these clarifications, suitable strategies and methods for handling complexity can be selected.

Figure 6.1 shows six sequential steps, which can be used as a guideline for implementing an adequate method for managing a complex challenge. This guideline describes the general logical sequence of steps, but does not represent a strict process. With each step through the guideline one obtains more information about the considered complex system. This knowledge can also require iterations of previous steps. For

M. Maurer, *Complexity Management in Engineering Design – a Primer*,
DOI 10.1007/978-3-662-53448-9_6

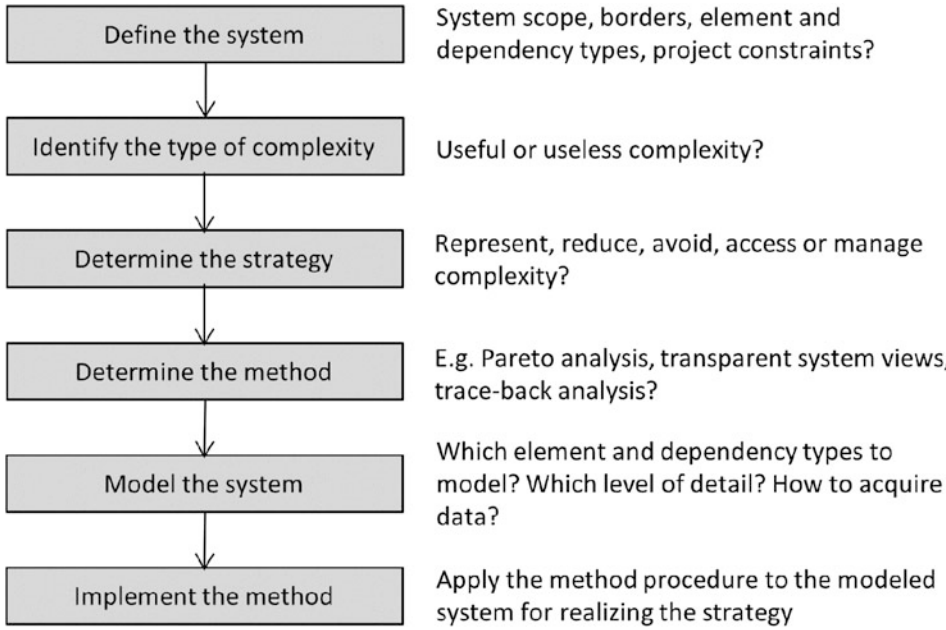

Fig. 6.1 Systematic approach on implementing complexity management for a system

example, it may become necessary to improve the system definition once the system modeling reveals ambiguities or missing clarity of element classification.

The guideline starts with the task of defining the system. This is required for identifying the origin of observed complexity. Next the type of complexity needs to be determined, e.g. if it is useful or useless in terms of the higher management objective. Before selecting a specific complexity management method the strategy has to be determined. This can be based on the type of complexity and the time scope of the management approach (from short to long term). Once the method of complexity management is specified, the system can be modeled adequately for the final implementation. In the following sections, the steps visualized in Fig. 6.1 are described in detail. Special attention is paid to the task of information acquisition, as this represents a difficult and laborious task, which heavily impacts the entire process outcome.

Complexity appears in many systems and contexts, which explains the variety of viewpoints, definitions and methods for handling complexity [1]. Examples clarifying the steps of this guideline have been chosen from the field of structural complexity. The reason therefore is that this kind of complexity and associated models and methods are applicable and widely used in many engineering fields. Structural complexity management [2] represents one specific view on complexity and has been applied on many challenges concerning technical, process-based or organizational networks [3, 4]. From this perspective, system elements and their dependencies form the core of a complex system. Senge and

Stermann describe this definition of complexity as detailed complexity (in contrast to dynamic complexity) [5, 6]. In structural complexity management not only the number of system elements and dependencies, but also their resulting constellations serve as input for analyses [7, 8]. Several system characteristics and the behavior can be conducted from these analyses [4]. The abstract character of the guideline in Fig. 6.1, however allows for the application of any kind of complexity modeling.

6.1 System Definition

Dealing with a complex engineering challenge often implies that the problem has not been understood completely. Thus, an instant system analysis seems not to be helpful on such a basis of incomplete information. Consequently, the initial task needs to be defining the system, as it helps clarifying the scope of the considered problem, system boundaries, considered system components and general objectives.

The initial question of the system definition should be about the origin of complexity. It is important to mention that consequences of complexity and its initial source can be located in different places. For example, one can observe extreme difficulties in managing the assembly process of product components in order to build a variety of products. However, the observed complexity (complexity impact) can be the result of a non-optimal portfolio of product components. That would mean that the origin of complexity is located in the area of product components, while negative impact can be observed in the process areas. If one would not question the origin of complexity, one would likely tend to create a process model and take measures of assembly optimization, and the origin of complexity would not even be part of the model.

A subsequent second question should be for the potential impact of complexity in the considered system. This is important as the pure existence of complexity is not a sufficient reason to take measures. For example, while complex effects happen during the combustion process in a car engine, this does not impact a person while driving. Even if the complex processes are located inside the car, they are not part of and nor linked to the driver's system. Consequently, this complexity does not impact the driver's system.

In the case when complexity has impact to a system in question, this leads directly to a subsequent question about the potential harmfulness of complexity. In many systems it is easy to identify a kind of complexity. For example, more detailed modeling increases the number of system elements and their interconnections—resulting in more structural complexity. However, the pure existence of complexity in a system does not necessarily require any countermeasures, if the effects resulting from this complexity are not harmful. A harmful effect could for example be increased uncertainty about component impacts within a product, which then leads to numerous unexpected and cost-intensive constructive changes to product components.

If system complexity with harmful, undesired impact could be identified, the next question should be about the objective of a complexity analysis. Reduction or avoidance

Table 6.1 Questions for clarifying the system definition

#	Question	Examples
1	Where does complexity in the system emerge from?	Dependencies between components or process steps
2	Which effects result from system complexity?	Unexpected change propagations, unexpected process delays
3	Which effects are harmful?	Expensive and time-demanding adaptations
4	What is the objective of complexity analysis? Which results are expected?	Possibilities for the improved development of a product

of complexity per se cannot be an objective. Benefits can only be achieved by decreasing negative or harmful impacts, which are the consequence of existing complexity. Defining the desired project results helps to specify clear objectives and makes project progress and success measurable. Table 6.1 summarizes the introductory questions for clarifying the system definition. Guided by these questions, the general system elements and dependency types, levels of detail, system states and boundaries should be specified.

6.2 Identify the Type of Complexity

With a definition of the considered complex system on hand, the next step is to identify the type of observed complexity. This step partly overlaps with the system definition, which already specifies the source area of complexity (and distinguishes it from the area of complexity impact). Lindemann et al. distinguish four main areas of complexity in engineering design: market, product, process and organizational complexity [2]. This classification can be helpful, as for each area of complexity different approaches of management have been developed and established, e.g. product modularization [9] in the product area or process streamlining in the process area [10]. Depending on the specific challenge, a different set of categories can be helpful too. For example, Ashkenas looks at complexity from an organizational perspective and defines managerial behavior, process evolution, product and service proliferation and structural mitosis as the four sources of complexity in organizations [11].

In addition to this classification, complexity can be described by characterizing it as useless or useful. An easy example explains this: If an enterprise produces customized production plants, significant complexity could emerge from the large number of customer requirements, functions and components. And this quantity can create complexity, for example if technical adaptations are required and component interdependencies result in undesired change impact. Also the large number of components can cause significant process complexity, for example when managing customized product assembly. If in this example, the step of system definition resulted in identifying the source of complexity as being the numerous requirements, functions and components, then complexity could be

reduced by standardizing the so far customized plants. In fact, standardization is a common strategy for complexity reduction. However, while this strategy would reduce the complexity, in this use case it would definitely also influence the company's market offer negatively.

In this simplified example useful complexity has been reduced, resulting in a negative effect to the company. In fact, in many cases complexity contributes to a competitive advantage, if this complexity can be handled. Then complexity is useful, and even increasing complexity could be beneficial—providing that this complexity can still be managed. A common example for such an increase of complexity would be the further enlargement of a product portfolio. With such a step companies intend to cover more market niches and increase the company's success.

Anderson describes several cases for useful complexity in his book "The long tail" [12]. He explains that enlarging the product portfolio even with low-selling products can be beneficial in today's economy. As positive examples he quotes Amazon with its tremendous amount of offered products. However, it has to be mentioned that Anderson describes the business models of companies offering mutually independent products via Internet stores. If one product is added to the portfolio the complexity does not increase significantly, as products do not affect each other (logistic considerations are neglected in this example). However, integrating an additional variant of a component in a car could have highly complex effects, as this would affect the whole system due to the large number of dependencies between components. This is the reason why Anderson's concept of "selling less of more" so far works for non-interconnected product portfolios only. But it shows that the potential of managing larger amounts of complexity could result in increased market strength. In other words, being able to manage useful complexity and therefore being able to increase it can create competitive advantages.

After answering the questions in the system definition step and considerations about the usefulness of the observed complexity in question, one should have a solid basis for selecting an appropriate complexity management strategy.

6.3 Strategies and Associated Methods for Handling Complexity

In many situations observed complexity appears as a negative system characteristic, which people dislike being confronted with. Thus people tend to implicitly chose the strategy of reducing complexity. But the explicit selection of an appropriate strategy for approaching complexity challenges is important—and therefore represents a step in the complexity management framework introduced here.

Available resources and the time frame for required results are the main criteria for selecting an appropriate strategy. Figure 6.2 links the three strategies with their time scope (adapted from [13]). In general, strategies for handling complexity can be classified into three groups: avoidance, reduction and management.

Reduction of complexity can be applied as a short-term measure for minimizing already existing complexity [14]. If, for example, the offering of a complex product spectrum with

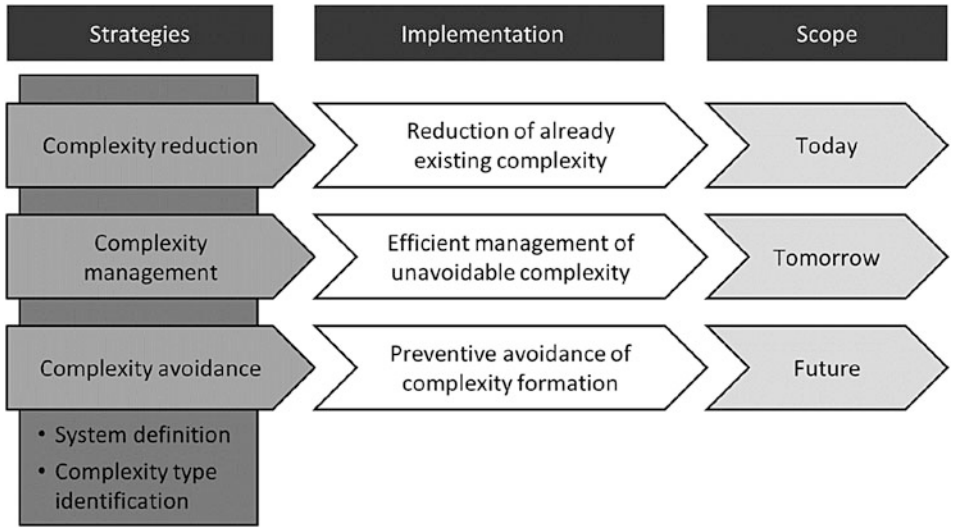

Fig. 6.2 Approaches towards handling complexity (adapted from [13])

extensive component variety results in process and production failures, reducing the product variety instantly reduces complexity. However, one must be aware of the fact that such a reduction is not going to the root of the problem; instead it only attenuates the consequences. For this reason a strategy of complexity reduction should be accompanied by more comprehensive complexity handling measures once the urgent need is satisfied. For the example of the complex product portfolio, the root cause of the large variety of offered products could be a sales concept that accepts too many customization requests. Thus, if the offered product portfolio is only reduced once and no further measures are taken, complexity would increase again over time.

As mentioned above, complexity can be viewed as a threatening experience. So it is understandable that complexity reduction is often seen as a generally useful measure. However, also beneficial types of complexity exist, because it leads to a competitive advantage. If, for example, a large product portfolio offer is decisive for market success, then the reduction of associated complexity would result in a negative impact, i.e. decrease the competitiveness of the company [2]. Therefore it is important to identify the type of complexity (useful or useless) before selecting a strategy of complexity management.

Management (controlling) of complexity is a strategy for handling unavoidable internal complexity, which arose through external sources (and therefore is unavoidable) [13, 14]. This strategy includes measures for complexity handling in development work and process design. An example could be the implementation of a configuration tool that supports the assembly of customized products, or a knowledge management concept that supports development work by increasing the reuse of existing solutions. As the implementation of such measures requires significant effort, this complexity management is a

strategy with at least mid-term time scope. The term unavoidable may create the impression that this type of complexity is undesired and useless in terms of the company's success. Therefore it needs to be mentioned that when complexity is useful in terms of a company's competitiveness, this complexity should also not be reduced or avoided—and therefore it is unavoidable.

Complexity avoidance is a strategy oriented towards the future [13]. It represents the consequent focus on the root causes of complexity emergence. This strategy is not useful as an ad hoc approach, as it is aiming at preventive avoidance and therefore has no effect on the status quo. In addition, measures of complexity avoidance typically require a significant effort in implementation and therefore aim at a long-term scope. In terms of the product portfolio example, possible measures could be the implementation of entrance barriers for the creation of new product components. For example, if the sales department requests the development of a new component, then one decision criterion could be a minimum purchase quantity. A certain threshold of purchase orders then needs to be met before development work is initiated. This could avoid the uncontrolled inflation of the product portfolio by component variants that only get ordered once. This and similar measures of complexity avoidance often require significant changes of business processes and even a company's culture.

Especially if companies do not have complexity management established permanently into their processes, they notice the need when complex problems negatively influence their daily business. In this situation, measures of complexity reduction and control have to be applied first, but should be accompanied by introducing complexity management for future success. The fundamental changes required for many approaches of complexity avoidance (as well as comprehensive complexity management/control approaches) need forward-looking planning and cannot be implemented as ad hoc solutions in crisis situations.

The necessity of treating useful and useless complexity differently has been addressed above. In addition, it needs to be mentioned that successful complexity management can turn initially harmful complexity into useful complexity. As long as a huge product portfolio cannot be controlled, process and production failures can be harmful to the company. And a company could be tempted to reduce product components and variants—even if a broad market offer would confer competitive advantages. The better the means of complexity management, the more complexity a company can support and apply for its success. Consequentially, the increase of useful complexity should also be named as viable strategy for handling complexity, assuming that the adequate management tools are on hand.

It should be noticed that useful complexity does not only have to be located in the product domain (as described in the example of a complex product portfolio), but could also emerge from complex processes that serve the customer demands. Examples could be advanced possibilities of product configuration and fast delivery by means of optimized business processes.

Also the relocation of complexity can be observed in practice: In the early years of mobile communication, Nokia became one of the dominant players in the market with an impressively large product portfolio [15]. Managing such a portfolio with short product life cycles is definitely challenging. But the variety guaranteed Nokia's strong position in the market. Then with the advent smartphones, Apple Inc. entered the market with its iPhone product in 2007. Even though it was only a single product variant, in 2008 customization became possible by installing a personal selection of (third-party) apps. Building up an ecosystem of software applications, assuring their functionality and safety and managing payment services created new challenges (and opportunities). So the useful, market-relevant complexity associated with a large product portfolio was shifted from hardware to software products.

Another strategy of complexity management is the adequate representation of the complex issues. As already introduced, complexity means the lack of transparency. If a complex system, its elements and dependencies can be visualized, then the resulting easier access to the system increases system understanding and possibilities of system interaction. Especially if several people are required for solving the problem, a suitable visualization can facilitate the discussions and prevent misunderstandings.

Methods provide procedures that support systematic problem solving. Many methods aim at managing complexity, with different approaches. Selecting an appropriate method for solving a complex problem can be challenging, because the ambiguity of complexity hinders clear determination of a method. In other words, it is difficult to select the right tool without knowing what to fix. Application of the guideline shown in Fig. 6.1 shall help minimize the uncertainty when selecting an appropriate method. This systematic approach also clarifies the risk of selecting a method without sufficient previous analysis of the complex problem. In complex, non-transparent situations people tend to stick to methods they are familiar with—especially if time pressure is an issue; however, this does not imply that the method is suitable for the specific challenge. The following sections describe some methods for handling complexity depending on the selected strategy.

6.3.1 Create Transparency by System Views

As a lack of transparency is a main characteristic of complex problems, it seems likely that comprehensible system views can facilitate solving them. Especially when system knowledge is shared by several experts, a comprehensive system view can focus the effort on a common target. Methods for creating transparent system views aim at element and dependency acquisition, preparation of system views and the enablement of interactions of experts with those views.

A large variety of representations exist for modeling complex problems. For the creation of transparent system views highly sophisticated approaches are less useful, as they do not provide easy access for many people, especially if they come with different knowledge backgrounds.

Graph notations have especially become very popular for easy-to-understand representations of system elements and their interdependencies. Reasons therefore are their simplicity and wide-ranging applicability. Computer-supported tools use different mechanisms for element alignment (e.g. force-directed graphs) and additional information representation (e.g. size, form or color of edges and nodes) for creating transparent system views, which often even allow for dynamic interactions [16, 17]. The requirements of big data analyses and progress in computational power bring up continuously improving graph representations with more and more functionalities [18]. But even if technical possibilities allow for representing an enormous amount of system details, it has to be taken into account that the human visual capacity is limited.

One solution to this limitation is the application of specific system views, which allow extracting relevant information from comprehensive system models in order to provide a manageable amount of information to a user. Transparency improves from highlighting only specific aspects of a system. This act of focusing on parts of the system can be done in two different ways:

The first possibility is to isolate specific parts of a complex system for representation. This means that all system parts that are not required for understanding the current question get neglected in this representation. If, for example, the collaboration within a large development department should be investigated, the visualization model could focus on the exchange of documents between employees. A graph representation would then allow the identification of centrally located employees as well as closely related employee groups. When this approach is selected it has to be assured that the highlighted system view permits one to draw meaningful conclusions. And it is important that the extraction of partial aspects does not provoke any misinterpretation, as they can easily appear: in the above example of an organizational structure the collaboration between employees could result not only from the document-based information flow, but also from informal communication, meeting structures, commonly developed components or the work on shared projects. If only one aspect of communication is extracted, accurate conclusions on the entire communication cannot be drawn.

The second possibility of creating transparency by highlighting specific aspects is to aggregate interdependencies into a system view. For example, dependencies like document exchange and component responsibility in a design department could be superimposed in order to create a general “dependency view”. While the specific reason for a dependency gets lost in such an aggregation, the density of represented information decreases while not neglecting any information. A detailed approach towards the creation of specific system views is described in the book Structural Complexity Management [2]. This approach is based on matrices as the basic tool for information acquisition and aggregation, whereas the visualization of resulting system structures can be done in matrix and graph format. Such system views provide better transparency for users than interacting with an unmanageable number of elements and dependencies in an unrestrained modeling of a complex system. In general, the creation of systems views should be only as detailed as necessary. It is not important which information is available, but which information is necessary to answer the question.

Lindemann et al. create system structure models based on two fundamental types of matrices, domain mapping matrices (DMMs) and design structure matrices (DSMs) [2]. A DMM is a matrix linking elements belonging to two different groups (domains), where typically each axis contains one group (domain) of elements. DSMs link elements belonging to the same group (domain); thus each element is shown on both axes of the matrix (the element order is identical on both axes). Eppinger and Browning provide details about the theoretical and practical application of these matrices [19].

Interestingly, all kinds of complex system structures can be built up based on the two matrix types DMM and DSM as their basic elements. Figure 6.3 shows the exemplary modeling scheme of dependencies between processes and documents. This scheme indicates the fact that executed processes deliver information to documents and that other processes require this information stored in documents. Both dependency types can be modeled using DMMs, as they link two different types of elements. A simple mathematical operation (matrix multiplication) then allows one to derive an aggregated network of processes indicating the information flow between them [4]. This aggregation suppresses the visualization of documents and integrates their links to processes into newly modeled links between process elements. While the fundamental DMMs are easy to acquire (the required information is often on hand), the aggregated process network provides an easy-to-understand, streamlined system view, which makes the process network transparent.

Figure 6.4 shows a more comprehensive modeling scheme of a system with product components that influence each other, people interacting with product components and

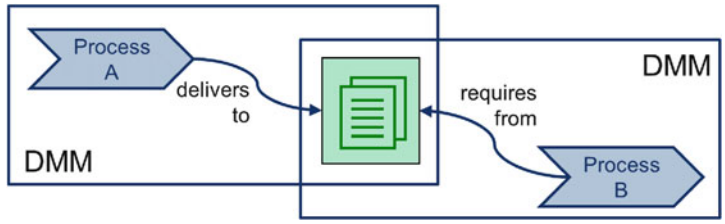

Fig. 6.3 Linking processes to documents by two DMMs

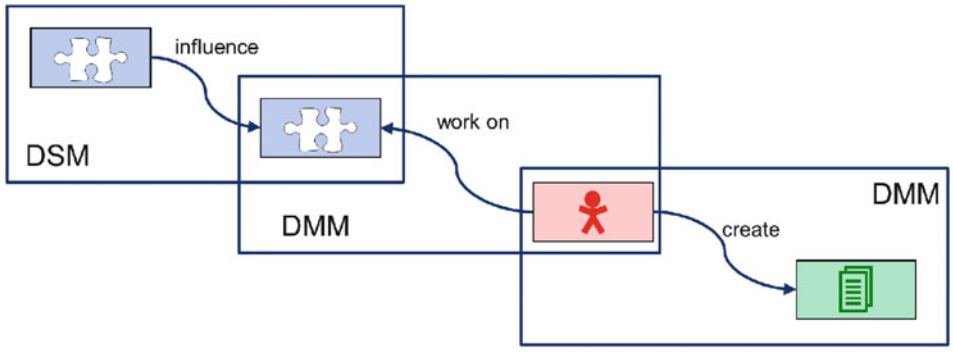

Fig. 6.4 Linking components, people and documents by DSM and DMM

documents created by people. This system can be decomposed into two DMMs and a DSM, as it is indicated by the rectangles in the figure. One aggregated view of this system could e.g. show mutual impact between people based on their work at (mutually linked) components and their common work on documents. In this case, components and documents would be suppressed and integrated into newly created links between people. Visualized as a transparent "impact map", this system view could be used for several decision-making processes.

6.3.2 Avoid or Reduce Complexity

In case of existing useless complexity, a strategy of complexity reduction can be applied. Reducing complexity is well-established for application on useless complexity of a system. Useless complexity means variants that increase costs and effort without adequate value in the market. Several methods in the engineering design field are applicable for implementing this strategy.

Especially variant management needs to be mentioned due to its significant relevance in the field. When aiming at the reduction of complexity, methods of variant management are applied for identifying product and component variants with undesired characteristics, e.g. low sales figures. If the removal of these identified variants does not impact other, desirable variants, the product portfolio can be revised accordingly. Identifying if and how components are interdependent in a large product portfolio represents a challenge, typically tackled by network analyses and intensive use of rule sets. Several authors provide comprehensive introductions to variant management and method sets for practical application [9, 20].

Obviously, reducing useful complexity cannot be a reasonable approach. However, many systems contain both useless and useful complexity, and it is important to reduce only useless complexity when applying this strategy. As well, one needs to keep in mind that so far useless complexity can be turned into being useful complexity if it can be controlled. Established evaluation methods can be applied for identifying useful and useless complexity. For example, a Pareto analysis facilitates the classification of products in terms of their profit generation and hence can separate useless from useful elements. Within a product portfolio, all products below a certain sales figure or above a certain production effort could be identified with this method. Value analyses or target costing approaches provide similar possibilities in terms of specifying the contribution of system elements to useful or useless complexity. In all these cases the selection of an appropriate evaluation criterion is of major importance.

Otherwise, product variants that add significant value (e.g. market strength) to the company at affordable costs represent the useful part of complexity. Schuh and Schwenk do not use the wording of useful and useless complexity, but their approach of optimizing the number of variants in a product portfolio means finding a subset with only useful complexity. It needs to be mentioned that their approach is not about reduction only.

Finding the optimal number of product variants can also mean extending complexity to fulfill untapped market potential [9].

Anderson explains that many markets ask for increasing product variety [12]. As the optimal number of product variants for an enterprise is determined by a tradeoff between the market value of variants and their costs (or complexity) to the enterprise, methods that decrease these costs can in return allow the increase of variants (useful complexity). One starting point can be reducing complexity in processes. Several authors describe methods for minimizing iterations, e.g. based on matrix approaches [7, 21].

Avoidance of complexity is a long-term strategy with the objective of not letting complexity emerge—and so no need arises to reduce it later on. Baldwin and Clark describe for product complexity that a method of decoupling product modules by interface design can help in avoiding complexity [22]. Such decoupling breaks up interdependencies between the modules and makes them mutually independent. Adaptations to one module then do not impact other modules and therefore avoids complexity resulting from change propagation.

For avoiding product portfolio complexity, defining entry barriers for the adoption of new product or component variants can be helpful. Companies, whose business models are not based on continuous product customization can efficiently avoid the rise of new variants. The selection of suitable evaluation criteria is of major importance. Wrongly chosen criteria can either be inefficient (so that undesired variants still emerge) or block meaningful variants from being introduced. That would mean that wrong criteria would hinder useful complexity from being built up.

6.3.3 Manage and Control Complexity

Here, managing complexity is understood as controlling it. Methods for controlling complexity primarily aim at the lack of system transparency as a main reason for negative impact from complexity. In highly networked systems then, decision making cannot be executed based on simple cause-and-effect chains, as they are either not accessible or, when extracted from the system, represent an incorrect simplification (neglect of side effects). The number of elements and dependencies in a complex system prevents one from acquiring as well as representing it completely. However, this is not even necessary. And it is important to notice that our daily decisions are hardly ever based on complete problem descriptions with all information on hand. In fact, successful decision-making is based on models, which contain only the relevant information. Thus, the challenge of managing complexity is not to acquire all the information, but to identify the relevant parts and to make them accessible in a suitable, comprehensible model. For this reason the early steps of system definition and identification of complexity types are of major importance.

Wildemann describes managing complexity as a strategy required for dealing with unavoidable complexity [13]. This can be misleading, as unavoidable still means that this complexity is undesired, but seems not to be removable from the system. However,

useful complexity is desired and contributes to an enterprise's market success. The better that useful complexity can be controlled, the more can be included in the system.

This can be explained with the example of an enterprise offering components for production automation. If the business model of the enterprise requires providing a multitude of components with high variability (e.g. different power ratings) and with the possibility to assemble different components into systems, then methods and tools for improving the configuration can be helpful to be implemented. Based on configuration rules, these methods and tools allow managing the products, variants and their possible combinations even on a large scale. Approaches on such internal configuration guarantee the development and integration of components and variants, which are compatible with the company's existing portfolio and extend it meaningfully.

Methods and tools for managing external configuration facilitate managing complexity at the interface to the customer. Customers can apply configuration managers for compiling and selecting customized solutions from a large, unclear spectrum of choice. Many configuration approaches try to make selections easier for the customer by letting him specify his needs instead of the technical details. The configuration tools then link the user requirements with the best matching product configuration. Felfernig et al. provide a profound introduction to the history and state of the art in configuration techniques and tools [23].

Schuh and Schwenk describe methods for matching internal complexity (number of component variants to be managed by the company) with external complexity (product variants visible to the customer) [9]. The objective is to realize a large external (useful) complexity with a small-enough-to-manage internal complexity. The better a company can manage this internal complexity, the more it can handle and consequentially extend the external market offer.

Other methods of managing and controlling complexity target the typical lack of understanding that goes along with complex systems. Daenzer and Huber describe checklists for complex system structures as a systems engineering approach, and Maurer explains them in the context of structural complexity management [4, 24]. An input checklist focuses on a specific system element (e.g. product component, process step or organizational resource) and indicates other elements that impact it [24]. This provides the possibility to monitor potential disruptive elements and foresee and control change propagation before it happens in an unregulated way. Input checklists can be applied beneficially on system elements that should not change, e.g. because of high effort, cost or safety issues related to such changes.

Feed-forward analyses are similar to input checklists, but do not serve for monitoring potential impactors but potentially affected system elements—emanating from one specific system element [4, 25]. Such an analysis allows estimating the consequences of adapting a specific system element by investigating the impact of this change to its surroundings. This perspective can be helpful when applied to those system elements, which are likely to be changed in the future (e.g. because customization requests are attached to these elements). If feed-forward analyses are already prepared for selected elements, once the specification of a change request is on hand, consequences can be systematically discussed and managed [4].

Trace-back analyses are meant for finding the element that initiates an observed system behavior or effect. Starting with the observed element, the system structure gets traced backwards until the initiating element is found [4, 25]. Depth-first search, breadth-first search, branch & bound and branch & cut are some methodical approaches to organize and optimize the search. Visual support, e.g. by graph representations, can also be very effective.

Daenzer and Huber introduce a simple but powerful influence portfolio, which allows one to identify the general embedding and parts of the behavior of selected system components [24]. For each system element, the outgoing (active) dependencies to other system elements are indicated on the horizontal axis. On the vertical axis, the incoming (passive) dependencies from other system elements are noted. The relative location of each element in the diagram allows one to deduce the basic behavior of an element.

Figure 6.5 shows an influence portfolio for the main components of an automotive airbag system [2]. The diagram is divided into four areas. In the lower-right area, elements with mainly outgoing and only few incoming dependencies are located. Those are active

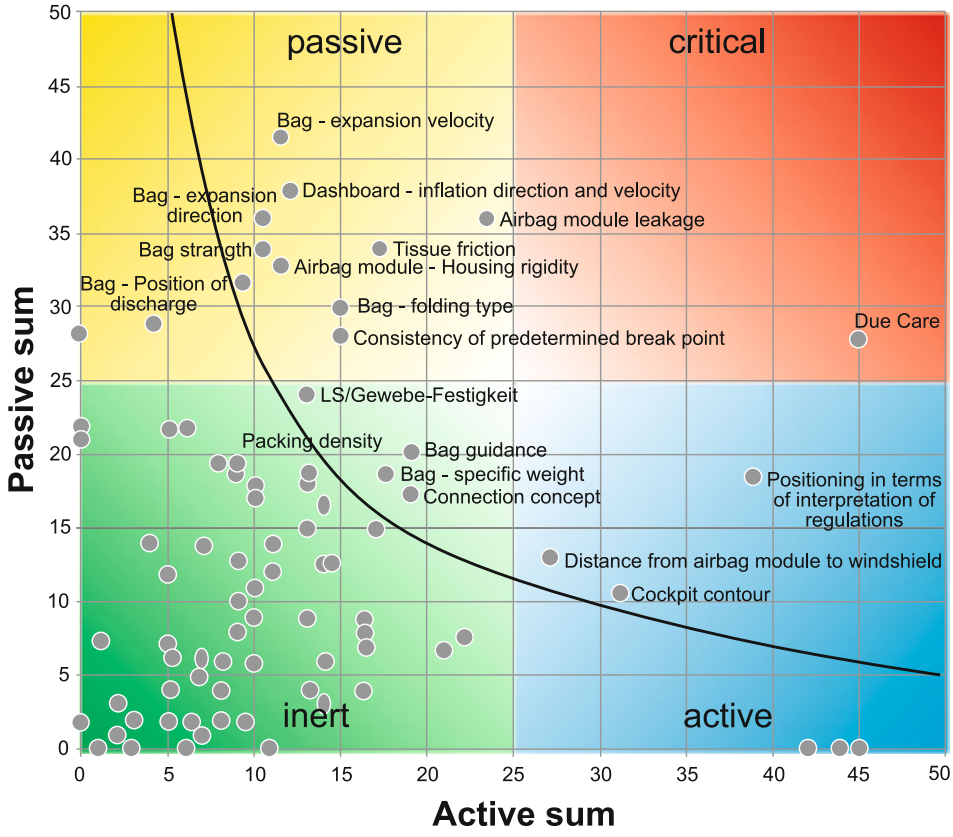

Fig. 6.5 Influence portfolio for an airbag system [2]

elements, which can influence the whole system via their many connections, but do not get influenced much from other elements. Elements located in the upper left corner behave contrary and are called passive elements. These elements possess many incoming and only few outgoing dependencies. Elements in the lower left corner are call inert elements, as they generally do not possess many connections to the system and therefore do not influence and are not influenced by the system very much. Elements located in the upper right corner represent the so-called critical elements and they possess the most dependencies within the system, having incoming as well as outgoing dependencies. Because of the many dependencies it is very likely that these critical elements are involved in any kind of changes to the system.

The criticality of an element is computed by multiplying the number of outgoing (active) dependencies with the number of incoming (passive) dependencies. The curve shown in Fig. 6.5 represents a line with constant criticality. When applying the impact diagram, typically such a line is selected as a threshold for identifying these elements, which need to be more closely investigated or monitored because of their high criticality. Of course, setting a value for this line is a subjective decision. In the project that created the diagram shown in Fig. 6.5, all elements with a higher criticality than indicated by the line became listed on a checklist, which had to be discussed for possible problems whenever a significant change anywhere in the airbag system was planned. As well, planned changes to any one of these highly critical elements had to be approved by the project management.

Methods similar to impact analyses, feed-forward and trace-back analyses are known by other names, e.g. cause-and-effect analyses. All these methods are based on modeling the complex system with nodes and edges in network form. These methods are meant for systematically managing changes to the system by increasing transparency and reducing uncertainty even when working with systems consisting of a large number of elements and dependencies.

Product modularization is often applied for managing and controlling complexity in large product portfolios [9]. From a system structure perspective, modularization means bundling highly interlinked product parts into modules and combining these modules by standardized interfaces, which means fewer and easier to control dependencies. In the same way as modularization, platforms or building blocks as well as differential and integral product design represent useful methods of managing useful complexity [4, 22].

Creating transparent system views, as explained in Sect. 6.3.1, is also a powerful technique to support controlling complexity. It allows people to interact with a complex system, detect anomalies, experiment with if-then-scenarios and discuss possible measures. This can be explained using the visualization of a student race car in Fig. 6.6 (adapted from [4]). The force-directed graph shows 27 main components with their interdependencies. The alignment of elements intuitively uncovers some basic system characteristics. These can be interpreted and applied for managing system interactions.

One central element (frame) connects three main subsets. The central embedding and the many dependencies of this element indicate its critical importance to the system. Three elements (gear shift, data logging and wiring harness) are isolated from the other parts of

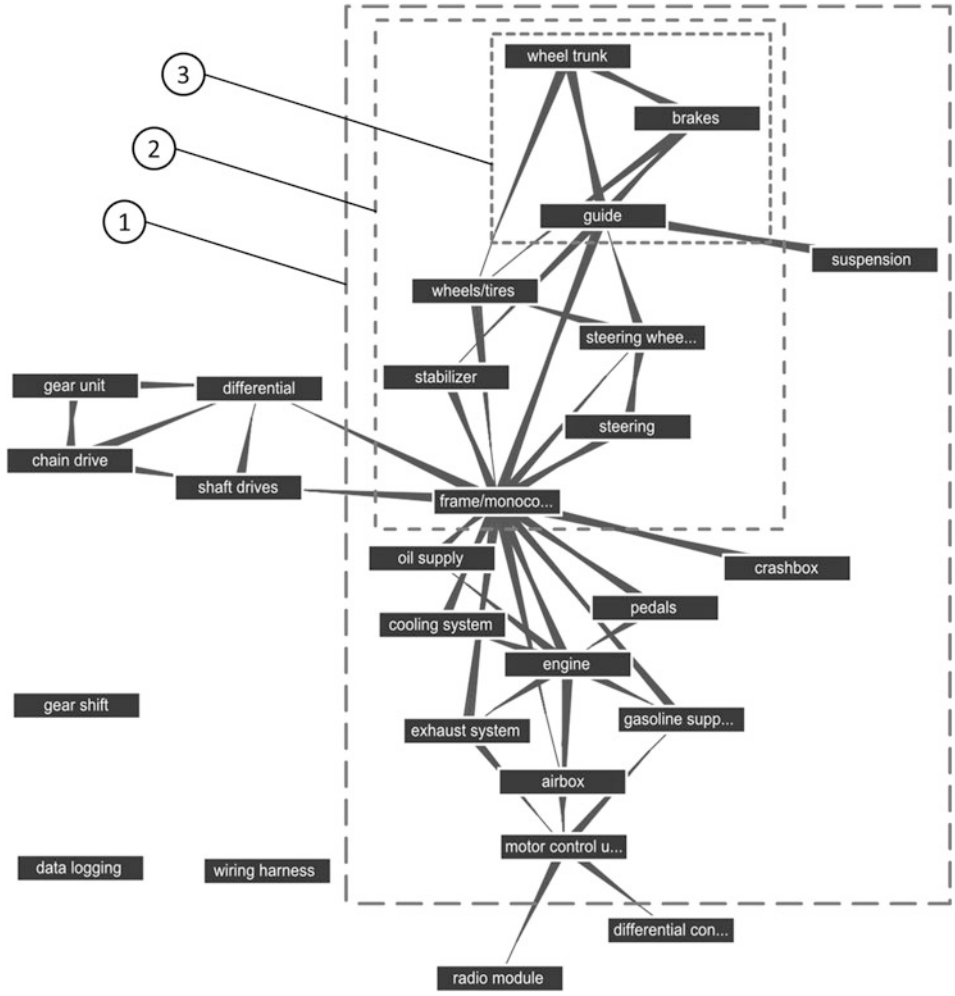

Fig. 6.6 Transparent system structure for facilitating complexity management (adapted from [4])

the system; also four elements (suspension, crashbox, differential control and radio module) represent leaves, which are only connected to one other element of the system. This minimal connection to the system makes it easy to control impact to and from these elements.

Basic tool-enabled graph analysis can facilitate further interpretation of the structure. All elements located within the frame indicated with 1 make part of the same strongly connected component. This constellation is defined as a set of elements, which are (indirectly) connected to each other by at least one path. From a control and management perspective, it is relevant that strongly connected components encapsulate feedback loops. So, cyclic effects as described by system dynamics models can only occur within this constellation.

In densely connected system structures, strongly connected components can become very large. And further subdivision of the structure can be helpful. The frame indicated with 2 in Fig. 6.6 contains elements forming a bi-connected component, also called a block. In this constellation all elements are mutually (at least indirectly) connected by two separate paths. Thus the integral character of a block is significantly higher than in a strongly connected component. Because of the two paths between each element, for example possible change impact cannot be avoided by simple, single measures. Engineering work at physical components forming a block structure require intensive organizational collaboration. An intensification of the block is indicated by 3 in Fig. 6.6. This constellation is called a complete cluster or clique; all elements are directly connected to each other and adapting any one of them always requires consideration of all other elements.

In general, interpretation of structure visualizations can support the control and management of complex systems. In this context it is worth mentioning that structural interpretations make part of peoples' daily life. We identify bottlenecks in processes or resource allocations and so put an interpretation to an articulation node, as it is called in graph theory. And when looking at an organization we talk about a key person, when in a graph depiction this person would be centrally embedded to the structure with many interdependencies to other parts of the system. In an impact diagram, such a person would be characterized as highly critical to the system.

6.4 System Modeling: The Challenge of Information Acquisition

Representations and analyses of complex system structures often use matrix approaches. Lindemann et al. describe a generic five-step approach on structural complexity management using matrices, which are complemented by graph representations [2]. Especially the task of information acquisition is challenging, because "the strength of the result and interpretation depend greatly on the reliability and validity of the data and the assumptions" [26]. Furthermore, information acquisition can be extremely resource-demanding, which can lead to problems of fatigue or training effects [2, 27].

Several requirements have to be considered in order to model the relevant system information. With those in mind, a suitable method of information acquisition has to be selected and applied. This application can be impeded by several possible errors, which can yield inaccurate models. Therefore, requirements, methods and possible errors will be described in the next sections.

6.4.1 Requirements for Information Acquisition

Information acquisition represents an important task in any approach on structural complexity management. In this context, several authors mention requirements, which have to

be fulfilled for reaching high-quality system structures. In the following paragraphs these requirements are aggregated and assessed concerning the need for methodical support.

For information acquisition using a design structure matrix (DSM), Dong explains the importance of adequate quality and completeness of information acquisition with its direct impact to subsequent tasks like analysis or interpretation [28]. Whereas the quality of input information and analysis output correlates positively, a negative correlation can be stated between increasing input quality and economization of resources. "Since the DSM is a tool to analyze the design project and to seek improvement, it is important that the data is accurate. When necessary one has to trade the speed of data collection with the quality of the data" [28].

The quality and completeness of structural information is difficult to measure. Dong refers to the interaction density ratio as an indicator for system model quality based on the number of system elements and dependencies [28]. If a system structure is represented in a DSM, this ratio compares the total number of off-diagonal marks with the total number of rows in a DSM [29]. Dong hypothesizes that an interaction density ratio of 6 indicates that enough information about a system has been acquired [29]. Maurer proposes the use of structural constellations (e.g. articulation nodes or star-shaped structures) for plausibility checks during an acquisition process [4]. These plausibility checks can also be based on the existence of direct and indirect dependencies between elements, and make criteria for deciding about the need for iterative information acquisition [30].

Bartolomei mentions that the reason for a lack of quality of a system structure can result from focusing on technical knowledge only [26]: "The engineering domain is methodologically ill-equipped to describe and represent components beyond the technical domain, such as describing the factors that influenced design decisions, mapping social interactions, and understanding systems processes. This is problematic for systems-level modelling frameworks as they are designed to represent knowledge that spans the social and technical domains" [26].

6.4.1.1 Adequate Level of Detail

Modelling system elements and dependencies too abstractly will only provide trivial insight; too detailed and the models can hardly be acquired within a given time and budget. For this reason an adequate level of detail is of major importance for successfully conducting information acquisition. Bartolomei criticizes a model describing a jet turbine with only 60 elements [26]. The author argues that the model of a highly complex system cannot provide significant benefit if it gets simplified too much. In contrast to this, Browning discusses resource problems arising from too extensive system structures [3]. He proposes an aggregation of system elements, i.e. modelling the system in less detail when the system size exceeds manageable amounts. Dong aggregates elements by defining three general segments: social, technical and natural [29]. She declares that dependencies within and between all three subsystems have to be acquired for obtaining a complete model. In hierarchical order, Dong defines component-level knowledge as the detail-level knowledge, which can be acquired from experts. However, system-level

knowledge is required for describing complex systems and needs to be assembled from the component-level knowledge of several experts [29].

Biedermann et al. also mention the importance of an adequate level of detail and suggest top-down modelling of system structures [31]. In this approach, dependencies are only acquired between top-level elements in the beginning. Next, possible dependencies at the detail level are investigated. Such dependencies can only exist if their superior (high-level) elements have already been linked. This means that large numbers of possible dependencies at the detail level can be excluded from further consideration if no high-level dependency has been acquired first. With this procedure Biedermann et al. try to enable modelling at a detail-level while keeping required resources at a minimum.

Not only system elements, but also dependency descriptions define a system's level of detail. Rowles mentions that the application of dependency weighting (instead of modelling the existence/non-existence only) can be unfavourable if interviewed experts cannot provide reliable information [32]. This means that the applied level of detail and the available system information have to match.

6.4.1.2 Accessibility, Traceability and Extensibility

Lindemann et al. describe that information acquisition processes can be highly iterative [2]. This is why acquired system elements and dependencies need to be accessible at any time. Dong indicates the DSM as being favourable, as "DSM acts like a browser that provides user directions to use the existing information database" [28]. Reasons for the declaration of elements and dependencies should be documented, as this can help in retracing decisions made in earlier acquisition processes. Dong highlights the importance of up-to-date documentation of all system elements in case of later system extension [29].

Especially for the modelling of large structures, the application of databases can facilitate the interaction with the model. For example, Ahmadi et al. describes the acquisition of a product design process by using a database [33]. Accessibility, traceability and extensibility are directly connected with two further requirements: the need for a systematic procedure and appropriate methods and tools, which will be introduced next.

6.4.1.3 Systematic Procedure

A systematic procedure of acquisition is a precondition for later accessibility, traceability and extensibility of system structures. As well, adequate quality and completeness of structures only seem to be achievable with such a procedure. Eppinger describes a procedure based on the compilation of a DSM [34]. He explains that system elements have to be acquired first, followed by the dependencies. Eppinger and Salminen mention that knowledge owners should be interrogated about their required input: "It's important to focus on input rather than output because we have found that managers, engineers, and other product-development professionals are more accurate in identifying what they need to know than in describing what others need to know" [35].

Black et al. propose separating the acquisition of system elements from the subsequent acquisition of dependencies [36]. This procedure allows one to only ask knowledge owners

about dependencies between elements under their own responsibility. As structure acquisition is time-consuming, this way every expert only gets burdened with the minimum of dependencies. Another approach is the separate acquisition of organizational and product dependencies. Both systems can then be compared to each other and the results may lead to iterative structure acquisition [32]. Pimmler and Eppinger propose an interrogation scheme, where information provided by knowledge owners is documented together with initially formulated estimations [37].

Some authors see the need for more detailed acquisition processes [28, 38]. Avnet describes a four-step procedure on information acquisition, which includes an intensive mix of acquisition techniques [38]. Observations of team meetings, surveys and interviews get combined and applied to a verification process by a team leader. Dong introduces a seven-step approach, which separates the collection of system elements and dependencies from subsequent system documentation [28]. Knowledge owners are interrogated by the system modeller, but they are not involved in formulating the system dependencies in the model afterwards. Bartolomei proposes a procedure called "qualitative knowledge construction", which consists of eight sequential steps [26]. This procedure applies "thick data", as mentioned in the grounded theory [39]. Sabbaghian et al. apply software for assuring the systematic execution of information acquisition [40]. The authors criticize other methods of information acquisition (i.e. interviews and meeting participation) as being too resource-demanding and too difficult to coordinate.

The examination of requirements for structure acquisition shows that high-quality information has to be generated at the right level of detail. Information needs to be accessible at any time and should be acquired in a systematic process. Methods for fulfilling these requirements will be described in the next section.

6.4.2 Methods of Information Acquisition

A detailed survey and classification of applied acquisition methods allows us to conclude the topic described above of fulfilling acquisition requirements. Almost 100 publications could be identified which contain structure acquisition by expert interrogation. In most of these publications, the act of conducting interrogation was only mentioned without any further information about applied processes or methods. Only a few contributions describe methods, advantages and disadvantages in detail. These publications have been examined and six basic methods of information acquisition can be extracted.

The investigation of scientific publications in terms of the origin of presented system structures shows that information acquisition is only considered sufficiently in a few cases. This means that the correctness of many analyses, interpretations or optimizations could be doubted, because the quality of applied structures is often unclear.

From over 500 investigated contributions to journals and conferences (all dealing with system structures) almost two-fifths do not describe any conducted information acquisition. Of course, one cannot conclude that in these projects information acquisition was not

executed. But obviously this step and its documentation was not of primary importance in these contributions. Now, without clear statements about the input information and their acquisition, it is difficult to evaluate the results. Another one-fifth of the investigated contributions avoided the effort of information acquisition by either applying system structures provided by other publications or creating exemplary system structures themselves. In another one-fifth of the surveyed publications, structural information is extracted from data sets, documents or models. Here the authors make use of a highly effective form of information acquisition. Especially if the import of data can be automated, large-scale structures can be obtained without extensive acquisition effort.

If the acquisition of system structures requires the interrogation of knowledge owners, then methods like workshops or interviews are required. Such methods were applied in one-fifth of the publications, and then the methods in these publications were examined in greater detail. Unfortunately, most authors who mention acquisition by interviews or workshops do not provide specifications of the acquired systems; therefore it is hardly possible to assess the effectiveness of approaches described.

Only seven publications mention the size of the system that had been acquired. Here, the numbers of considered system elements varies from 18 [41] to 600 elements [42]. Only six authors specify the duration of the acquisition process, which varies from a few hours [43] to several months [44]. Details about the number of knowledge owners involved could only be identified in eight contributions. Numbers vary from 6 [43] to 70 people [10].

In general, acquisition processes are rarely described. And it seems reasonable to suppose that the lack of descriptions goes along with insufficient consideration of information acquisition. As the acquisition of a system structure represent the basic input for subsequent modelling, analysis and interpretation, available methods and challenging barriers have to be known. Methods can be classified into six groups: analog survey, digital survey, documentation, observations, interview and estimations. In the following paragraphs these methods will be detailed. Table 6.2 sums up advantages and disadvantages of each method.

6.4.2.1 Analog and Digital Surveys

An analog survey means to investigate documents which have been filled out by system experts. Multiple-choice questions are commonly applied, because they are easy to process for the interviewee and the evaluation can be automated. An obvious advantage of analog surveys is time efficiency, as many experts can be interrogated without much effort [28]. Dong also mentions possibilities of statistical analysis as advantageous. Browning acquires the development process of an aerospace company using an analog survey [45]. He observed that experts have different understandings of the questions asked, which can influence the resulting quality negatively. In addition, he mentions that incomplete responses to the questionnaire can be problematic. Rowles describes analog surveys for acquiring the system model of a jet engine and the associated organization [32]. Similar to Browning, he describes the risk of a low response rate as disadvantageous. Avnet let the participants of design sessions fill out analog survey forms after each session [38]. He, as

Table 6.2 Methods for the acquisition of system structures

Methods	Advantages	Disadvantages
Analog survey	Time-saving [28] Possibility of statistical analysis [28]	Incomplete response to questionnaires [45] Different understanding of concepts [45] Random assessment of quantifiers [45] Multiple-choice constricts creativity and solution space [38] Information loss by constricted responses [28] No justification of responses [28] Specification of target-state instead of as-is-state [28] A priori problem [26] Low response rate [32]
Digital survey	Possibility to process large quantity of data [40] Requirements on resources minimized [40] Possibility to model dynamics over time [40]	Unclear terminology leads to rework [40] Responsible interaction of several people required [40]
Documentation	Low effort, automatable [2]	Outdated data [28, 45] Unrealistic data [28] One-sided representation [28]
Observations	Possibility of fast conflict resolution [28] Understanding of situations facilitated [38] Interpretations of surveys or interviews [38]	Potential information loss because of hierarchy and social pressure [38] Subjectivity of observations [38]
Interview	Identification of direct dependencies [28] Identification and resolution of disagreement [28]	Danger of intentional concealment of the real process [26] Time-consuming process if DSM gets filled out during interview [38]
Estimations	Minimum effort of acquisition [45]	Doubtful data

well as Dong, states that especially multiple-choice questions constrict possible responses, which can lead to information loss [28, 38]. Furthermore, Dong mentions that experts may tend to describe a target state instead of the state as-is, and then responses can hardly be justified. And Bartolomei describes the "a priori problem" as being a critical challenge [26]. This problem means that the act of information acquisition itself may constrain a system by creating inappropriate assumptions through the framing of questions.

In general, analog surveys are designed for the interrogation of many people with affordable effort. The success of such surveys highly depends on the preparation of questions and the motivation and commitment of the participants. This is because the completion of forms often cannot be controlled. Even if the effort required for conducting

analog surveys is relatively low, it increases with the number of participating people and the quantity of question rounds.

Today, digital surveys are widely applied and services like SurveyMonkey (www.surveymonkey.com) are easy to apply even for first-time users. Digital surveys overcome the boundaries of conventional analog surveys. These boundaries are set by the number of interrogated people, the amount and complexity of considered data and the possibilities of automated evaluation. Unfortunately, digital surveys often seem to be even less binding for participants than analog surveys. This can result in very low response rates from participants.

6.4.2.2 Documentation

Often, structure information can be extracted from existing documents. Lindemann et al. identified several relevant sources, e.g. project management charts or the documentation of design methods like quality function deployment (QFD) or TRIZ [2]. Dong applies requirement lists [28], Avnet collects information from project management software [38] and Browning extracts elements and dependencies from organigrams [45]. If applicable, this method of information acquisition is advantageous because of few required resources [2]. But the use of outdated or unrealistic data for information acquisition can be disadvantageous [28, 45].

6.4.2.3 Observation

An observation is a methodical approach on information acquisition, where the knowledge owner is not directly occupied with the formulation of dependencies. The system modeller attends the daily workflow of experts or participates in meetings or workshops. Then the modeller creates the structure based on the insights gained. For example, Avnet took part in design sessions as a passive observer [38]. He describes that observations help increase the system understanding. For this reason, observations can be preliminary work for later interpretation of surveys and interviews [38]. And the direct integration of system modellers into the workflow of knowledge owners can enforce fast conflict solving [28]. It must be mentioned that observations can be demanding for the system modeller, and he needs to be a semi-expert. Avnet describes possible information loss as a disadvantage of observations, which can result from the organizational hierarchy and social pressure in observed meetings. And the system modeller must be aware of his own subjectivity in interpreting the observations made [38].

6.4.2.4 Interview

In many publications, interviews are mentioned as a methodical approach for information acquisition. Dong applies interviews in two case studies in the automotive industry [28, 29] and Black et al. develop a DSM describing the development process of automotive brakes [36]. Rowles applies interviews for pre- and post-processing a survey on the development of a jet engine [32]. Also Browning and Sabbaghian et al. use interviews for the preparation of a subsequent survey [40, 45]. Whereas many authors mention the application of

interviews, only few methodical details are described. One of these descriptions is given by Bartolomei, who mentions the formulation of open questions for his approach on qualitative knowledge construction [26]. Intensive research has been undertaken concerning interview types and their application. For example, Kvale provides a comprehensive overview of planning, conducting, analysing and validating different types of interviews [46].

In the information acquisition of system structures, the methodical application of interviews needs to be increased. Interviews seem to enable the reliable identification of direct dependencies and the simple resolution of disagreements [28]. However, Bartolomei mentions the danger of intentional concealment by the interviewed experts to be difficult to identify [26]. And Avnet mentions the disadvantage of a time-consuming interview process if dependencies are documented during interviews [38].

6.4.2.5 Estimations

In general, system modellers can complement information, if not provided by experts or other resources during the acquisition. This can become necessary if acquiring information from the right sources would be too costly. If technical values are on hand, sometimes interpolation can be applied. If not, estimations can be useful. Browning describes a situation where knowledge owners were mandatory but did not participate in the survey. Browning estimated lacking information in order to be able to establish the system structure [45]. It has to be mentioned that reliable estimations can ask for expert knowledge. And uncertainty of the decisions made must be taken into account.

The six methods presented in the paragraphs above all possess specific advantages and disadvantages. These are aggregated in Table 6.2. For the conduction of high-quality structure acquisitions some authors combine two or even more methods. Especially interviews are often applied for validating information acquired by surveys. In these cases interviews are not used for entire system modelling, but for quality improvements only, as interviews represent the most resource-demanding method. In this context, Dong mentions that surveys result in worse results than interviews [28]. Also the use of available documentation always needs to be considered before applying other, more resource-consuming methods. In general, the specific project situation asks for the application of adequate methods. Decision parameters are mainly the system size, number of knowledge owners, available resources and required quality of resulting structures.

6.4.3 Barriers Against Successful Information Acquisition

When interacting with complex systems it is useful to be aware of the typical mistakes. Dörner explains failures people make when interacting with complex systems [47]. All of these failures can be directly related to a lack of managing a system's interdependencies. Dörner states that people often do not analyze problems adequately and therefore the

concepts of problem solving are only based on small subsets of the entire system; large numbers of elements and their connectivity to the system are then neglected.

Often systems get considered as being uncoupled aggregations of subsets. One reason for that can be a person's own core of knowledge. Simply said, an electrical engineer will likely tend to search for a solution in an electrical (sub)set of the system than in a mechanical or software set. However in the case of such a one-sided problem consideration, if changes happen to non-considered system parts then the system behavior appears to be unpredictable.

Another typical mistake when dealing with complex systems is the insufficient consideration of side effects. For example, this can happen when single key figures get used for assessing complexity. As complex challenges go along with non-transparency, the desire for a simple assessment possibility becomes understandable. Unfortunately, only very specific complex system perspectives can be rated by a condensed value without neglecting significant aspects. In computer science complex problems can be differentiated by their degree of computational difficulty [1]. The required computing time or the minimal size of required computational code (Kolmogorov complexity) are used as metrics, assuming that the problem can be mathematically formulated (see Sect. 3.2). However, complex systems often contain unknown interdependencies which cannot be described by mathematical equations, and simple complexity indicators only take a small part of the system into consideration. For example, a popular system complexity rating is by its number of components; they are easy to identify and count. However, this complexity assessment does not consider interdependencies, so integrated versus modular product designs are not rated differently by this approach.

If undesired effects result from interacting with complex systems, then people tend to oversteer in order to correct these effects. The system's non-transparency, however, does often not allow one to predict the impact from these chosen measures, which can—because of manifold interdependencies—bring up further undesired effects. And it often seems to be impossible to systematically search for a solution to a complex problem, because limited resources seem to be insufficient for investigating a large-sized system. In such a situation, tendencies towards authoritarian decision making can often be seen, even if a sound decision basis is non-existent.

Unmanaged, uncontrolled complexity can result in a lack of decision-making abilities, incorrect decisions with significant negative impact, frequent changes due to lack of sustainability or long process durations. But as introduced in Sect. 6.3.3, successful management of complexity enables tremendous opportunities for enterprises. For example, a large number of product variants can be developed, maintained and offered to the market if the mutual interdependencies can be handled. Or more customer requirements can be realized when complex processes are well-controlled—which means that iterations are unlikely and change impact is predictable. The benefits of controlled complexity are so significant that many successful enterprises work with systems just at the edge of manageability.

The methods applied for acquiring system information (see Sect. 6.4.2) are designed for achieving high-quality information at manageable effort and take initial requirements (see Sect. 6.4.1) into account. Several barriers exist in application, which make system information acquisition a challenging task. Next, these barriers are introduced, because awareness of existing barriers is mandatory for future improvements in information acquisition.

Figure 6.7 indicates ten barriers to successful information acquisition and associates them with a basic product design process. These barriers have been documented by authors dealing with requirements and methods for information acquisition. In the upper part of Fig. 6.7, the result of general process steps of product design is depicted: product specification, functional structures, principal solutions, module structure and preliminary layout. Information acquisition about system structures is typically conducted while structuring the modules and creating the preliminary system layout. Knowledge owners contribute to this

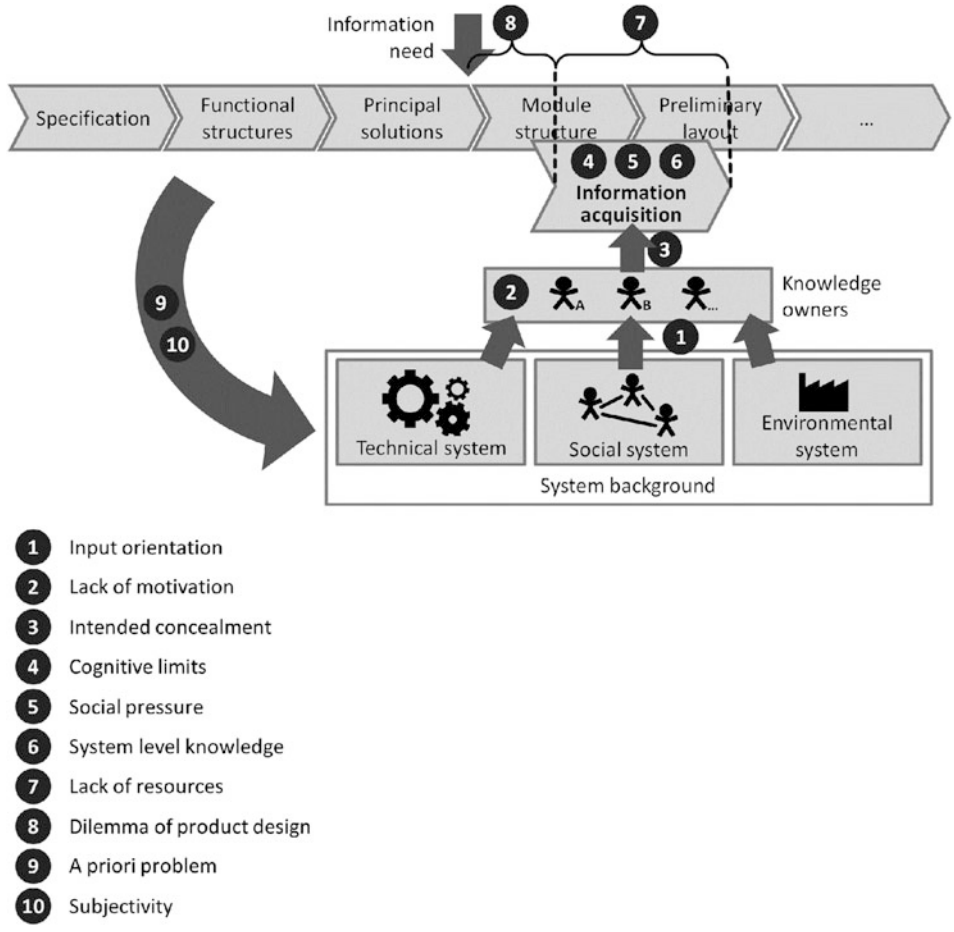

Fig. 6.7 Barriers to successful structure acquisition, appearance in the development process

information acquisition using their knowledge background about the technical, social and environmental systems. The ten barriers are explained in the paragraphs below.

6.4.3.1 Input Orientation

Browning as well as Eppinger state that people's thinking is input-oriented [34, 45]. "It's important to focus on input rather than output because we have found that managers, engineers, and other product-development professionals are more accurate in identifying what they need to know than in describing what others need to know" [34]. As the input for one expert represents the output of another one, information has to be merged in order to achieve a consistent structure. The use of different wording can make such merging difficult. In Fig. 6.7, this barrier is located between the knowledge owners and their system background, as the input orientation can hardly be influenced by knowledge owners actively.

6.4.3.2 Lack of Motivation

It may happen that interrogated knowledge owners are not sufficiently motivated to provide the required system information [32, 45]. One possible reason is that the knowledge owners cannot see the benefit for themselves and may interpret the interrogation as unprofitable extra work. Furthermore, they may see the interrogation as documenting their job role concretely so they can be replaced more easily by another worker. Support from higher management will often be helpful [32]. However, while emphasizing the importance of this task, this does not solve the problem of unseen benefits. Specific interview techniques and early project involvement of experts could help increase the motivation [46].

6.4.3.3 Intentional Concealment

Bartolomei mentions the risk of intentional concealment by knowledge owners as a possibly severe barrier [26]. Especially, if knowledge owners are not aware about the usage of information provided by them, their personal interest can influence the acquisition results negatively. In this context, Dong explains that knowledge owners tend to provide desired target-states instead of as-is states [28]. Plausibility checks and the interrogation of several knowledge owners can help to identify false information. However, this barrier should be resolved at the social level.

6.4.3.4 Cognitive Limits

Three barriers could be identified, which can appear directly in the information acquisition process (see numbers 4, 5 and 6 in Fig. 6.7): cognitive limits of interrogated experts, social pressure and the need for system-level knowledge. Dong mentions that the limits of human cognitive abilities have to be considered when managing complex systems [29]. Browning relates this limitation to the handling of DSM [48]. He notes that even system models containing a few elements only can overburden people. The barrier of cognitive limits could for example be resolved by representing only the required system information and to select appropriate, case-specific visualization techniques.

6.4.3.5 Social Pressure

Avnet as well as Dong describe that social pressure, for example resulting from workshop participants' places in the company hierarchy, can lead to information loss in acquisition processes [28, 38]. The effect is that statements made by higher-ranked participants are not criticized, or lower-ranked participants do not even impart their opinion. The barrier of social pressure is closely related to the prevailing meeting culture. If necessary, workshops participants should be at the same level in the hierarchy.

6.4.3.6 System-level Knowledge

System-level knowledge is mentioned by Dong as the compilation of many people's component-level knowledge [28]. The author describes that system-level knowledge is required for system analysis, but cannot be acquired from a single expert only; rather it must be compiled by the aggregation of many experts' component-level knowledge. Thus, it is important not to simply analyze structures at the component level and directly draw conclusions at the system level.

6.4.3.7 Lack of Resources

Whereas the above mentioned barriers that can occur in the acquisition process, lack of resources can impede the general process being conducted. Avnet describes shortening interviews, because the required engineers did not have time for extensive meetings [38]. However, time or resource constraints correlate negatively with the achievable information quality. Consequently, if resources are reduced the system structure should be acquired on a less detailed level. However, too abstract of a model may result in trivial analysis results.

6.4.3.8 Dilemma of Product Design

The earlier that changes and improvements are implemented in the design process, the easier (and cheaper) their realization gets. But earlier actions mean less knowledge (and more uncertainty) about the system in question. This is a general dilemma of product design, which is also valid for the application of system structures [29]. Structure acquisition is typically done after defining general system modules (see location in Fig. 6.7). However, knowledge about basic dependencies would be helpful for determining these modules—but the knowledge is not available at that time. Techniques of early evaluation of properties could help in overcoming this barrier in the future.

6.4.3.9 A Priori Problem and Subjectivity

The barriers of a priori problem and subjectivity are related to each other. The a priori problem means that assumptions brought into the information acquisition process by system modellers can constraint the resulting system. As these assumptions are not the result of well-founded information from system experts, they can falsify the model [26]. A typical example is the constriction of a survey's solution space by wrong assumptions implemented in the questionnaire. Such assumptions often result from the subjectivity of

system modellers. But not only system modellers, all participants in an information acquisition process bring in their own background [28, 32, 38]. This can impede the quality of resulting systems and can for example be counteracted by comparing information provided by several experts. In Fig. 6.7, the barriers of the a priori problem and subjectivity are depicted as impacts leading from the design process (i.e. from the daily work of experts) to the system background (i.e. experiences that shape personal knowledge).

6.5 Complexity Management Implementation

The objective of the final step within the complexity management framework is the method application to the system in order to realize the selected strategy. At this point of the process, the modelled system structures have been transferred into system knowledge; constraints and potentials of the system should be uncovered and meaningful interaction with it should be possible in order to improve the system. Improvement in this context refers to the initial questions answered during the step of system definition: What is the objective of the complexity analysis? Which results are expected? The acquired system knowledge needs to be transferred into actions for solving the complexity challenge.

The objectives are as manifold as the possible solutions and implementations are diverse. In general, the system understanding acquired in conducting the previous steps of the framework can be implemented on the complex system in two different ways. The first one aims at improving the interaction with a complex system so that complexity does not get reduced or avoided, but becomes more manageable. This can be realized by implementing possibilities of accessing, navigating and searching in the complex system structure, for example by means of adequate visualization and filtering. The second possibility of implementing the acquired system knowledge to the complex challenge is by improving the engineering system itself; this means to rearrange, eliminate or integrate new elements and dependencies within the system in order change the amount of complexity that has to be managed. For example, modularizing a so far monolithic product structure represents the implementation of this approach.

When implementing a solution one has to keep in mind that this is based on a system model, which represents an abstraction of the real system. Thus, even when the system modeling was conducted carefully, it might happen that the derived theoretical solutions do not represent viable options in practical application. Plausibility checks are mandatory before initiating the implementation.

In addition, one has to keep the importance of a holistic system view in mind when working on complex challenges. Each single measure elaborated in the selected solution approach might be easy and meaningful to implement in practice; however, different measures can influence each other when applied to the system. Thus, all measures have to be checked in their entirety for meaningfulness from a holistic system perspective.

The complexity management framework introduced in this chapter has been illustrated with examples from the field of structural complexity, because this concept is widely

applied in and is the fundament of many engineering methods. But the framework also matches with other modeling approaches. The very simple six-step approach is not meant to describe the exact process of solving complex engineering challenges. These processes are highly iterative, because knowledge about the non-transparent complex system can often only be obtained gradually. Thus, assumptions made initially in the process have to be revised when new knowledge is on hand. In fact, the presented framework shall present a guideline assuring that important steps towards the solution of a complex problem are not skipped and are done in the right sequence. While this seems to be obvious, practice shows that it is often not respected. The reason for that can be found in the typical failures made when interacting with complex systems (see Sect. 3.3), as they have been described by Dörner [47]. So it can happen that one selects a method for solving a complex problem even before an appropriate system definition is obtained. A person might be familiar with a specific method and the situation seems to require immediate action—thus well-known tools often become the first choice. For sure, this is not a promising approach. One has to be aware at all times that dealing with a complex engineering challenge means to be confronted with a decision based on incomplete system knowledge. So a stepwise, systematic clarification of causes, effects and objectives is required for reliably determining appropriate strategies, methods and implementations.

References

1. Mainzer, Klaus. 2008. *Komplexität*. Paderborn: Wilhelm Fink.
2. Lindemann, Udo, Maik Maurer, and Thomas Braun. 2009. *Structural Complexity Management—An Approach for the Field of Product Design*. Berlin: Springer http://medcontent.metapress.com/index/A65RM03P4874243N.pdf.
3. Browning, T.R. 2001. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management* 48(3): 292–306. doi:10.1109/17.946528.
4. Maurer, Maik S. 2007. *Strcutural Awareness in Complex Product Design*. Munich: Dr. Hut. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20070618-622288-1-1.
5. Senge, Peter M. 1994. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York: Doubleday.
6. Sterman, John D 2000. On an Approach to Techniques for the Analysis of the Structure of Large Systems of Equations. *SIAM Review* 4(4).
7. Kusiak, Andrew. 1999. *Engineering Design—Products, Processes and System*. San Diego: Academic Press.
8. Melnikov, O., V. Sarvanov, R. Tyshkevich, and V. Yemelichev. 1994. *Lectures on Graph Theory*. Mannheim: BI-Wissenschaftsverlag.
9. Schuh, Günther, and Urs Schwenk. 2001. *Produktkomplexität Managen. Strategien, Methoden, Tools*. München: Hanser Fachbuch.
10. Kreimeyer, Matthias F. 2009. A Structural Measurement System for Engineering Design Processes.
11. Ashkenas, Ron. 2013. *Simply Effective: How to Cut Through Complexity in Your Organization and Get Things Done*. Harward Business Press.

12. Anderson, Chris. 2008. *The Long Tail: Why the Future of Business is Selling Less of More*. Revised ed. New York: Hyperion.
13. Wildemann, H. 2008. *Komplexitätsmanagement in Vertrieb, Beschaffung, Produkt, Entwicklung Und Produktion*. 9th ed. München: Komplexitätsmanagement in Vertrieb, Beschaffung, Produkt, Entwicklung und Produktion.
14. Jania, T. 2004. Änderungsmanagement Auf Basis Eines Integrierten Prozess- Und Produktions-Modells Mit Dem Ziel Einer Durchgängigen Komplexitätsbewertung. Universität Paderborn.
15. Wikipedia.org. 2015. List of Nokia Products. https://en.wikipedia.org/wiki/List_of_Nokia_products#Mobile_phones.
16. Battista, G., P. Eades, R. Tamassia, and I.G. Tollis. 1999. *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ: Prentice Hall.
17. Lima, Manuel. 2011. *Visual Complexity—Mapping Patterns of Information*. New York: Princeton Architectural Press.
18. LaValle, Steve, Eric Lesser, Rebecca Shockley, Michael Hopkins, and Nina Kruschwitz. 2010. *Big Data, Analytics and the Path from Insights to Value*. MITSloan Management Review.
19. Eppinger, Steven D., and Tyson R. Browning. 2012. *Design Structure Matrix Methods and Applications*. Cambridge, MA: MIT Press.
20. Gießmann, Marco. 2010. Komplexitätsmanagement in Der Logistik. Kausalanalytische Untersuchung Zum Einfluss Der Beschaffungskomplexität Auf Den Logistikerfolg. BoD—Books on Demand.
21. Gebala, David A., and Steven D. Eppinger. 1991. Methods for Analyzing Design Procedures. In Proceedings of the ASME Third International Conference in Design Theory and Methodology, 227–233. Miami: ASME.
22. Baldwin, C.Y., and K.B. Clark. 2000. *Design Rules—The Power of Modularity.* Vol. 1. Cambridge, MA: MIT Press.
23. Felfernig, Alexander, Lothar Hotz, Claire Bagley, and Juha Tiihonen. 2014. *Knowledge-Based Configuration: From Research to Business Cases*. Waltham, MA: Morgan Kaufmann.
24. Daenzer, W.F., and F. Huber. 1999. *Systems Engineering: Methodik Und Praxis*. 10th ed. Zürich: Verl. Industrielle Organisation.
25. Hub, H. 1994. *Ganzheitliches Denken Im Management: Komplexe Aufgaben PC-gestützt lösen*. Wiesbaden: Gabler.
26. Bartolomei, J. 2007. Qualitative Knowledge Construction for Engineering Systems: Extending the Design Structure Matrix Methodology in Scope and Perspective. Engineering Systems Division. http://hdl.handle.net/1721.1/43855.
27. Alexander, C. 1964. *Notes on the Synthesis of Form*. Cambridge: Harvard University Press.
28. Dong, Qi. 1999. Representing Information Flow and Management in Product Design Using the Design Structure Matrix.
29. ———. 2002. Predicting and Managing System Interactions at Early Phase of the Product Development Process. Department of Mechanical Engineering, no. February 22:19–296.
30. Eichinger, Markus, Maik Maurer, Udo Pulm, and Udo Lindemann. 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: CD-ROM.
31. Biedermann, Wieland, Matthias Kreimeyer, and Udo Lindemann. 2009. Measurement System to Improve Data Acquisition Workshops. In 11th International Design Structure Matrix Conference, ed. Matthias Kreimeyer, Jonathan Maier, George Fadel, and Udo Lindemann. Greenville.
32. Rowles, C.M. 1999. System Integration Analysis of a Large Commercial Aircraft Engine. Masters Thesis in Engineering and Management. http://dspace.mit.edu/bitstream/handle/1721.1/9753/42769852.pdf?sequence=1.

33. Ahmadi, Reza, Thomas A. Roemer, and Robert H. Wang. 2001. Structuring Product Development Processes. *European Journal of Operational Research* 130(3): 539–558. doi:10.1016/S0377-2217(99)00412-9.
34. Eppinger, Steven D. 2001. Innovation at the Speed of Information. *Harvard Business Review* 79 (1): 149–158.
35. Eppinger, Steven D., and Vesa Salminen. 2001. Patterns of Product Development Interactions. In International Conference on Engineering Design ICED 01. Glasgow.
36. Black, Thomas A., Charles H. Fine, and Emanuel M. Sachs. 1990. A Method for Systems Design Using Precedence Relationships: An Application to Automotive Brake Systems.
37. Pimmler, Thomas U., and Steven D. Eppinger. 1994. Integration Analysis of Product Decompositions, no. September.
38. Avnet, Mark S. 2009. *Socio-Cognitive Analysis of Engineering System Design: Shared Knowledge, Process, and Product*. doi: 10.1016/j.anpedi.2008.08.013.
39. Glaser, Barney, and Anselm Strauss. 1967. *The Discovery of Grounded Theory*. Chicago: Aldine.
40. Sabbaghian, N., S. Eppinger, and E. Murman. 1998. Product Development Process Capture and Display Using Web-Based Technologies. *IEEE International Conference on Systems Man and Cybernetics* 3: 2664–2669. doi:10.1109/ICSMC.1998.725062.
41. Biedermann, Wieland, Ben Strelkow, Florian Klar, Udo Lindemann, and Michael F. Zaeh. 2010. Reducing Data Acquisition Effort by Hierarchical System Modelling. In Proceedings of the 12th International DSM Conference, ed. Matthias Kreimeyer and David Wynn. Cambridge, UK: Hanser.
42. Guivarch, Antoine. 2002. Car Engine Design Process and Organization Management Using DSMs. In 4th Design Structure Matrix Workshop, ed. Steven D. Eppinger, Daniel Whitney, and Ali Yassine. Cambridge, UK.
43. Sauser, Brian. 2006. Toward Mission Assurance: A Framework for Systems Engineering Management. *Systems Engineering* 9(3): 213–227.
44. Björnfot, A., and L. Stehn. 2007. A Design Structural Matrix Approach Displaying Structural and Assembly Requirements in Construction: A Timber Case Study. *Journal of Engineering Design* 18(2): 113–124.
45. Browning, Tyson R. 1996. Systematic IPT Integration in Lean Development Programs.
46. Kvale, S. 2008. *Doing Interviews*. London: Sage.
47. Dörner, Dietrich. 1997. *The Logic of Failure: Recognizing and Avoiding Error in Complex Situations*. Cambridge: Basic Books.
48. Browning, Tyson R. 2001. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management* 48(3): 292–306.