Introducing Complexity in Engineering

3

Complexity management requires a profound understanding of the matter of complexity. Therefore the general composition of complex systems will be introduced first, using examples to highlight important aspects and the large range of complexity. These examples are followed by a discussion of the difference between the meaning of the terms complex and complicated in the context of systems management. It will be explained why it is not just a minor linguistic difference, but a need for specific management approaches.

Despite the excessive use of the term complexity many of its definitions are either quite vague or very specific—and therefore only applicable to certain fields. For example, one mathematical definition of complexity (Kolmogorov complexity) is based on the minimal length of code required for generating a specific desired output [1]. Obviously, this is not helpful when dealing with the high-level development of an airplane or the management of a large infrastructure system. But even in engineering disciplines like systems engineering, a variety of definitions instead of a central common one can be identified [2]. Section 3.2 introduces relevant complexity definitions and indicated commonalities and differences. Special focus is placed on structural complexity, as it possesses major relevance for many applications.

Dörner mentioned that people tend to make specific errors when interacting with complex systems. For successfully managing such systems one must be aware of these mistakes, and one requires adequate methodical approaches even if the system in question seems to be non-transparent [3]. The typical problems occurring when interacting with a system are one specific consequence of complexity, which is explained in Sect. 3.3.

In the last section of Chap. 3, established engineering approaches towards complexity management are described. As each of them is covered by innumerable books, this section should only be a brief introduction, providing the basic understanding for later chapters. For example, Chap. 4 investigates the historic development of complexity management approaches. And Chap. 5 classifies the approaches, describes their differences and overlaps

M. Maurer, *Complexity Management in Engineering Design – a Primer*,
DOI 10.1007/978-3-662-53448-9_3

and links important contributors. While in Sect. 3.2 the definition of structural complexity is provided, Sect. 3.4.4 introduces dependency modeling, which gets applied in a manifold of approaches, methods and tools aiming at the management of this kind of complexity.

3.1 Composition of Complex Systems

When talking about engineering examples of complexity, often the development of the Airbus A380 gets mentioned. Especially in Europe this project has been in public awareness for a long period of time.

Without any doubt, significant technological challenges had to be met for planning and realizing the largest passenger airplane in the history of aviation. This challenge represented a new development, as it does not happen very often. Most development projects can be classified as change or adaptation developments, as they are largely based upon existing products. Here, representatively examples of new technical development challenges could be the integration of multiple new technical systems or the air conditioning for a large number of people. And as the integration of technical (sub)systems implies many interdependencies, small changes to one partial system can result in far-reaching, sometimes unpredictable and undesired consequences.

When mentioning complexity of the A380 development, first thoughts go to the technical product and its product complexity. And a technical product with such a huge scope definitely comprises much of this type of complexity. However, product complexity did not pose the only challenge in the A380 development: realizing the product required an adequate organization, e.g. with distributed development teams at several locations. The project size and the fact that it has been a new development resulted in a large organization size—and this organization formed a structure with numerous interdependencies, tremendous information flows and high dynamics.

The organization executes processes, e.g. the integration of a large number of customer requirements into the product. Since customer acquisition was of major importance for the new product, even in late stages of the development adaptations were still being conducted. And such integration of requirement-driven adaptations caused changes to the technical system and partly resulted in unforeseen impact (change propagation) because of the numerous interdependencies in the system. This impact resulted in laborious and costly rework, which also resulted in severe project delays.

Besides product-, organization- and process related complexity, another highly relevant source of complexity for the project was the embedding of the system into the environment it was designed to operate in. For example, passenger boarding processes have never before been designed for the large passenger capacity of an A380. Thus, procedures but also technical support systems (e.g. passenger bridges) had to be rethought and redesigned. And all these auxiliary processes and products have to be embedded into the airport system as a whole. A large number of interdependencies exist between the subsystems, which form the

Complex product
- Aerodynamics, position control
- Safety instructions
- Passenger comfort
- ...

Complex organization
- Distributed development team
- Several production plants
- ...

Complex embedding
- Passenger boarding
- Baggage and cargo handling
- Emergency cases
- ...

Complex process
- Integration of customer requirements at late process stages
- ...

Wiring in an Airbus A380*

Passenger bridges connected to an Airbus A380**

Fig. 3.1 Airbus A380 as a complex system, *Vitaly V. Kuzmin (http://vitalykuzmin.net/?q=node/248), CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0), via Wikimedia Commons; ** Hakilon, CC BY 3.0 (http://creativecommons.org/licenses/by/3.0), via Wikimedia Commons

greater airport system and have been impacted by changes required for embedding the A380 into it.

Figure 3.1 summarizes the mentioned aspects of complexity for the Airbus A380 project. Obviously, it would be easy to identify additional aspects of complexity; but also this exemplary description allows identifying one major characteristic of complexity: the large amount of interdependencies between system elements can produce unpredictable impact to the system (change propagation) when new system elements are implemented or existing elements or dependencies are adapted. For a person interacting with a system, complexity appears as a lack of logical match between inputs to and outputs from the system. Predictions about the system output based on input measures become hardly predictable, and control and management of such complex systems become challenging.

Another example of a complex system is illustrated in Fig. 3.2. Most people would agree that city infrastructure represents a complex system. However, the aspects that make the system complex are not always clear at first glance. In fact, infrastructure is a good example

Function of individual traffic

- Transportation of goods
- Passenger transportation
- Rescue operations

Extraordinary effects

- Spontaneous traffic jams
- Accidents and breakdowns
- Commuter/holiday traffic
- Mass events

Scope for design

- Design and layout of roads
- Signage
- Traffic light circuit

Large motorway junction*

Traffic jam on a motorway**

Fig. 3.2 Transportation as a complex system, *Bayerische Vermessungsverwaltung—http://www.geodaten.bayern.de (http://vermessung.bayern.de/open data), CC BY-SA 3.0 (http://creativecommons.org/licenses/by-sa/3.0), via Wikimedia Commons, **Alexander Blum (www.alexanderblum.de), via Wikimedia Commons

for a complex system of systems, which comprises for example energy, transportation, telecommunication and information, water and waste infrastructure systems. Those sub-systems are interconnected, and for example a failure in the energy system can lead to tremendous disruptions in the transportation and telecommunication systems.

When thinking about the complex aspects of a transportation system, often major motorway junctions with their confusing road layout come to mind. Figure 3.2 contains such a typical photograph. Even if the many interconnected roads cannot be comprehended in a first impression, this alone does not make the system complex. Roads and their junctions can be modeled as a structure of nodes and edges. And without further impact this structure is not characterized by any dynamics. Structural changes would include road construction (also inducing a low degree of dynamics) but also redirections, which can quickly change the usage of the structure. And this usage is decisive for the complexity of the transportation system. While the road structure remains mainly stable, fluctuations of traffic on individual roads induce high dynamics. Passenger traffic shows fluctuations daily, weekly as well as seasonally. And extraordinary, predictable effects like major cultural and

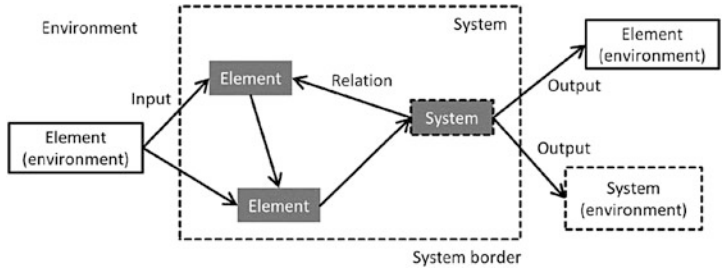


Fig. 3.3 System elements and interdependencies

sports events superimpose on those regular fluctuations. Unpredictable impact to the system occurs due to accidents, breakdowns or severe weather, which also burden the transportation system.

After introducing two real-world examples of complex systems, Fig. 3.3 depicts the basic composition of a system. It possesses a system border, separating the system itself from its environment. Inside the borders the system consists of elements, which are connected by interdependencies (relations). Section 3.4.4 will show that those elements and relations can be classified into groups by their meanings. Interdependencies also exist between elements of the system, and elements and systems in the environment outside of the system borders. The internal and external relations are decisive for the system's reaction to changes from the system's environment, either from external elements or other connected systems. In the other direction, a system produces changes that reach over the system border to impact external elements or systems.

This description of a system meets the definition of structural complexity, as introduced next in Sect. 3.2. And dynamic complexity requires system structures as a functional basis. Dynamics are not modeled in the generic system depiction of Fig. 3.3—but the interdependencies, which connect the elements, represent the paths along which dynamic effects and processes proceed. Knowing about the structure already provides information about possible dynamic behavior, e.g. because of feedback loops (which can result self-energizing effects) or bottlenecks (which can be critical because impact propagation gets channeled) [4]. System dynamics (see Sects. 3.4.3 and 4.3.5) builds upon system structures, as do many approaches and methods in systems engineering (see Sects. 3.4.2 and 4.3.4)—for example when designing system architectures or managing interfaces.

A system can also contain one or more other systems. In such a case one talks about a system of systems. In fact, both systems described above represent a system of systems. For example, the electrical system and the turbines form systems located within the entire system of the Airbus A380. Similarly, the roads and the guidance system (traffic lights, signs etc.) form their own systems within the general transportation system. For a comprehensive introduction to systems of systems, see e.g. [5].

The network built from elements and relations in Fig. 3.3 only indicates one static state of a system. As mentioned above, system dynamics models dynamic behavior based on such structures using additional modeling elements like stocks, flows and time delays. In addition, dynamic system complexity can emerge from changes to the existence of system elements and relations. In product portfolios for example, new components and possibilities of combining those components into new product variants can occur over time. And other components may disappear from the portfolio, e.g. because suppliers stopped production. In organizational structures the dynamic changes to system elements and relations can occur even more frequently and quickly than in technical systems. The collaboration and communication between employees in a company is constantly evolving, so that official organization structures often differ significantly from de facto structures. This has to be kept in mind when analyzing complex system structures.

Obviously the more relations and elements a system comprises of, the more non-transparent it becomes. Non-transparency in this context means that it is not clear how the system inputs are correlated with its outputs. The outputs cannot be predicted simply based on the inputs.

When applied to engineering tasks, the definition of the term complexity is often not very precise, compared to the way mathematical definitions are (see Sect. 3.2). In daily business, complex processes, products and organizations represent challenges in engineering. And even mixtures of these complexity domains meet, for example, in complex projects.

While most challenges in those engineering domains do not meet mathematical complexity definitions, people who have to interact with the systems experience that they do not understand the outcome based on their input to the system—and they call this as complexity. An example could be a situation where engineers are confronted with a product development process that led to extensive project time overruns when applied. As a countermeasure to those overruns, more resources could have been assigned to the process execution, expecting a reduction in process run time as consequence. If, however, the process time then would not decrease as expected (or would even increase) the engineers would experience a mismatch between system input and output, and would constitute a lack of system understanding. This exemplary development process appears to be complex to the person who is responsible for managing it. In other words, in the engineering context a system is often called complex if one cannot predict the system's output based on the given input.

The basis of complex engineering systems is the quantity of system elements and interdependencies—and on this basis, impact propagation and dynamic behavior takes place. The system elements can be classified into groups of similar objects often called system domains. Such domains can for example be process steps, product elements and organizational units. Interdependencies exist between elements within one domain as well as between elements of different domains. While identification and classification of system elements can help to improve system understanding, it is important not to reduce complex challenges to isolated system perspectives (see also Sect. 4.1.2 for the historical background on reductionism and Sect. 3.3 for typical failures when interacting with complex systems). Complexity results from the interaction in networks of elements.

In most modeling approaches, system elements receive close examination while the origin of system dependencies often gets neglected. However, many different dependency types appear in a complex system and it is important to differentiate between them for proper interpretation [4].

Distinction Between Complicated and Complex Systems

Especially in descriptions of industrial use cases, the terms complicated and complex often appear in the same context and are even used as synonyms. However, both terms represent different kinds of challenges and require specific methods for solving them. Incorrect characterization of a challenge and the subsequent mis-application of methods can be counterproductive. For this reason a more detailed consideration of complicatedness and complexity is helpful.

A typical example for a complicated challenge is searching for a needle in a haystack. This definitely represents a difficult, laborious task, but reaching the solution is only a question of effort. The more time and the more people work on the task, the higher the probability of a faster solution. The task can be parallelized so that several people search smaller haystacks. In general formulation that means that a complicated problem can be divided into smaller and less complicated problems, which can be processed independently. Maurer (according to Ehrlenspiel) mentions that the individual capability is linked to the complicatedness of a task stating that "the term complicated system describes the subjective difficulty in interaction with technical systems that often depends on one's personal abilities". Ehrlenspiel indicates that "identical situations can be complicated for one person, but not for another" [4] according to [6].

In contrast to the "needle in haystack problem", the forecasting of world climate represents a complex challenge. Such challenges are characterized by high dynamics and are difficult to subdivide into smaller, more manageable tasks. The reason therefore is that the elements of the complex system are highly interrelated. Development of water temperatures in the ocean is related to cloud formation, air movement, rainfall etc. Inadequate simplification by extracting specific aspects can easily lead to wrong system models, as important system impacts and outputs get neglected. In contrary, as high as the haystack hiding the needle may be, the system elements do not possess any relevant interdependencies—and therefore the system can be subdivided.

Thus, the large amount of elements interconnected by relations is a significant characteristic of a complex system, where the emphasis lies more on the relations than elements. It is not purely the number of product components, process steps or organizational roles that cause complex system behavior, but instead it is their degree of mutual connectivity. A dense interconnectivity of elements makes a system non-transparent to a person interacting with it and results in momentum of that system. And though as high as a hypothetical haystack may be, the system is transparent as the task, inputs and outputs are easy to understand, and system reactions to user interactions are predictable.

Complicated and complex challenges call for application of different approaches and methods. One approach towards complicated problems has already been mentioned:

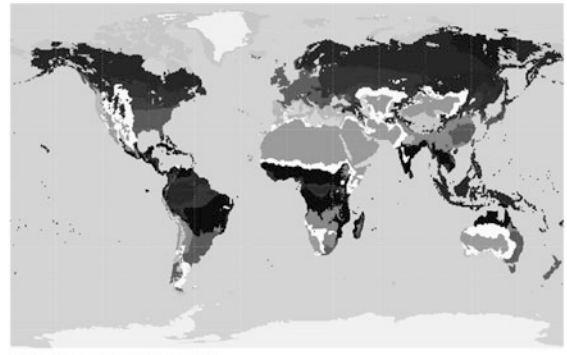
increasing applied resources. As a complicated challenge can be subdivided into smaller, independent work packages, more resources can accelerate finding a solution.

Interestingly, the same approach can cause additional problems when applied to a complex challenge. In general, it is a good idea to use more resources; in the use case of the world climate, different people are assigned to model and analyze water temperatures, wind, cloud formation etc. But because all these aspects of the complex challenge are interrelated, the pieces of work are interrelated too, resulting in the new creation of organizational complexity. The engineering discipline of systems engineering tackles such challenges, where a complex technical problem has to be considered as a more holistic challenge with additional process and organizational complexity as the consequence of work distribution.

For the example of the world climate one could argue that simplified forecasting models exist. Furthermore, regional subsystems are created for forecasting on country or city basis. That would mean that reductionism can be successfully applied. In fact, reducing the modeling of a complex system goes along with the risk of neglecting relevant aspects—which then can result in wrong analysis and prognosis results. The variation between different climate predictions in practice and the deviation between assumptions of former models and experienced climate in reality gives good evidence of this effect.

The examples mentioned above show the importance of correct classification of problems as being complicated or complex, because incorrect classification can lead to wrong measures being taken. In fact, it is a common mistake to counteract the appearance of a complex problem with increased resources—without preparing to manage the increase in process and organizational complexity. Figure 3.4 summarizes the general distinctions

A pile of hay roles, now find the needle...*



Climate map of the world

Complicatedness

- Problem subjectively difficult
- Solution reachable by „hard work"
- Characteristic of system perception

Complexity

- System irreducible
- Unforeseen developments
- System characteristic

Fig. 3.4 Complex versus complicated systems, *Scott Bauer, U.S. Department of Agriculture, via Wikimedia Commons

between complicatedness and complexity. And the following sections provide definitions and layout types as well as approaches towards the management of complexity.

3.2 Complexity Definitions

Even though the term "complex" has become common word for describing many situations, its correct meaning is not easy to describe due to many aspects and different perspectives. However, if complexity shall be managed it is important to understand its origins, relevance and impacts.

The term complex originates from the Latin word stems com which means together, and plectere which means weave. Thus the combination complex can be translated as interwoven. Ashby states that "[. . .] there are complex systems that just do not allow the varying of only one factor at a time—they are so dynamic and interconnected that the alteration of one factor immediately acts as cause to evoke alterations in others, perhaps in a great many others" [7]. This description of complexity already contains the most relevant aspect: because of interconnections between elements in a dynamically behaving system, simple one-to-one effect chains rarely exist. And so changes to one element can result in avalanche-like impacts.

Scientific fields dealing with complexity have different perspectives, which results in a variety of definitions of complexity. Even within the engineering field not all aspects of complexity are commonly shared and standardized definitions that serve all fields of application do not exist [8].

The perhaps most precise definitions can be found in mathematics and computer science, because they permit the deduction of exact complexity measures. With these measures two systems can be directly compared in terms of their complexity and thresholds that trigger specific actions can be set. The required computing time or the minimal size of software code (Kolmogorov complexity) can be used as complexity measure, if the problem can be mathematically formulated [9]. Unfortunately, this kind of complexity definition and measure is not applicable to systems, which cannot be fully modeled by mathematical means. For typical complex engineering systems this is the case.

Mainzer categorizes different types of complexity from a mathematical and computer science perspective [1]. He aggregates the determination of complexity by code size or computation time as computable complexity. Besides this, notions of information complexity and dynamic complexity exist (see Fig. 3.5).

Information complexity (entropy) comprises the phenomena of noise, describing the fact that parameters oscillate with no clear timely behavior. Depending on the frequency, those oscillating effects are named white, pink, red and gray noise. Noise effects are often illustrated by the sound that results from electrical oscillations in an amplifier. However, the same effect occurs in many other situations and systems, e.g. with stock prizes or car traffic

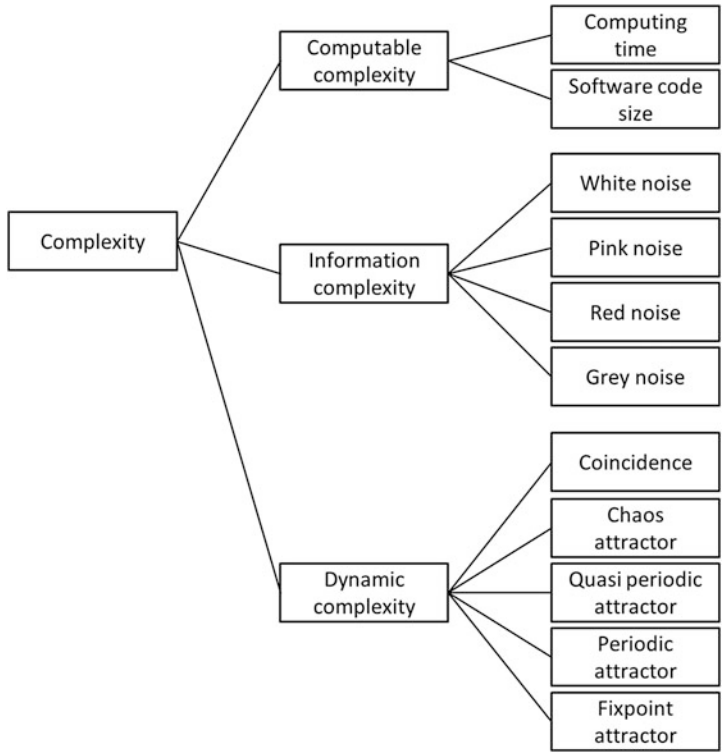


Fig. 3.5 Classification of complexity from a mathematical/computer science perspective [1]

on highways [10]. Mainzer explains: "1/f b spectra [describing the cure of a periodic signal] represent the pattern for distinguishing the different forms of signal noise in the world. [. . .] Signals of time series also provide hints towards self-organizing complex structures [. . .] [and] secular trends [. . .]. Time series analyses with 1/f b spectra are independent from specific systems and [. . .] can be applied to all kinds of dynamic systems" [1]. A well-grounded introduction to noise effects and their relevance for complexity is given by [10].

Degrees of dynamic complexity can be identified depending on the attractor, which is applicable for a complex system. An attractor is a state a dynamic system gets "attracted to" in the long run. A fixed-point attractor represents a state of equilibrium that remains unchanged. Non-linear complex systems can also reach periodically changing equilibrium states (periodic and quasi-periodic attractors) as well as turbulent or even random states (chaos attractors, coincidence) [1].

After introducing his classification of complexity types, Mainzer highlights that this should not be seen as an approach of reductionism (see also Sect. 3.2). "The structures of complex systems cannot be reduced to their single elements, but can only be explained by their collective interaction" [1]. This picks up Aristotle's famous statement that "whole is greater than the part" (Aristotle, cf. Euclid, Elements, Book I, Common notion 5).

Herbert A. Simon, a Nobel Prize laureate in 1978, defines a complex system as "one made up of a large number of parts that interact in a nonsimple way. In such systems, the whole is more than the sum of the parts, not in an ultimate, metaphysical sense, but in the important pragmatic sense that, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole" [11]. Simon describes the significance of a hierarchical system structure for complex systems, saying that a hierarchic system is "a system that is composed of interrelated subsystems, each of the latter being, in turn, hierarchic in structure until we reach some lowest level of elementary subsystem".

Simon's parable of the two watchmakers became a famous and often cited exemplification for the benefits of hierarchical structures for the evolution of complex systems. In this parable, both watchmakers build complex mechanical watches composing of 1000 parts each. One watchmaker (Hora) architects his watches based on 111 subassemblies on three levels, with each subassembly consisting of ten parts. On the lowest level the 1000 basic parts are arranged in 100 assemblies. Those 100 assemblies are further aggregated into ten higher-level assemblies, which then form the entire watch. The other watchmaker (Tempus) builds the whole product as one single assembly of 1000 parts. Now it is assumed that every time a watchmaker has to interrupt his work (e.g. for taking a phone call) the currently unfinished assembly falls apart and has to be reassembled. With a simple quantitative analysis, Simon shows that Tempus loses much more work when being interrupted and that for him the probability for successfully finishing the assembly of a watch is ridiculously low compared to Hora [11].

Simon transfers the findings from his parable into the evolution of complex systems stating that "the time required for the evolution of a complex form from simple elements depends critically on the numbers and distribution of potential intermediate stable forms". So, "complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not. The resulting complex form in the former case will be hierarchic" [11].

Concerning the dynamics of complex systems, Simon explains that "hierarchies have a property, near-decomposability, that greatly simplifies their behavior". Near-decomposability means that "interactions among the subsystems are weak, but not negligible" and "intra-component linkages are generally stronger than inter-component linkages. This fact has the effect of separating the high-frequency dynamics of a hierarchy—involving the internal structure of the components—from the low-frequency dynamics—involving interaction among components" [11].

Additionally, in terms of comprehending complex systems, Simon states that "empirically, a large proportion of the complex systems we observe in nature exhibit hierarchic structure." Furthermore, "if there are important systems in the world that are complex without being hierarchic, they may to a considerable extent escape our observation and our understanding" [11].

A basic classification of engineering complexity is shown in Fig. 3.6. Market complexity can be seen as a major source of complex challenges, because market conditions and adaptations can hardly be influenced by enterprises. Market complexity can result for

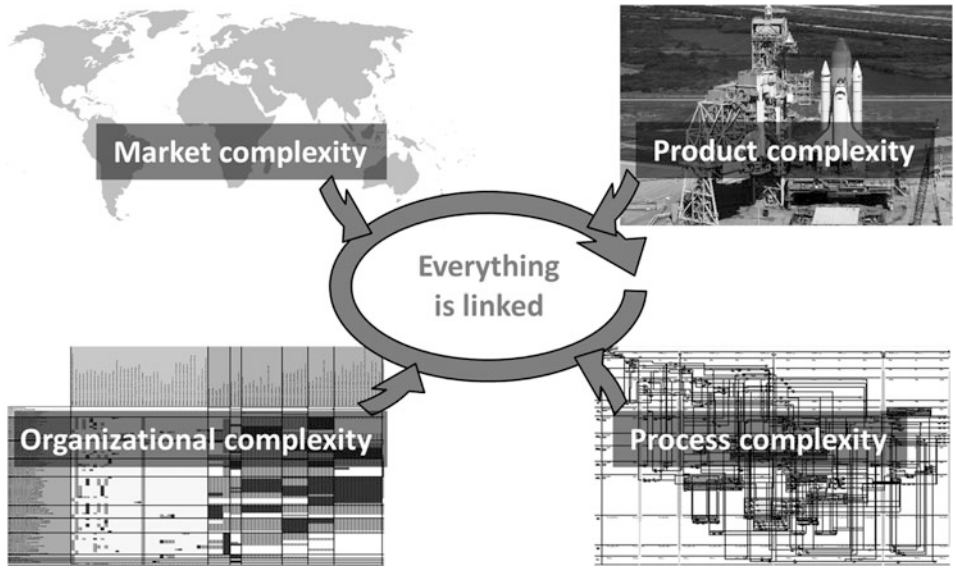


Fig. 3.6 Complexity fields in engineering, NASA/Frank Michaux, via Wikimedia Commons

example from a large variety of customer requirements that have to be fulfilled. In addition, laws, regulations or regional and linguistic peculiarities can create boundary conditions that add to the market complexity. From a company's point of view market complexity is also called external complexity, because of its origin outside of the company's direct influence.

The external complexity (as seen from the company's perspective) is faced by an internal complexity that results from the company's product portfolio. This includes combination possibilities among the variety of components, which lead to product specifications that shall fulfill the external complexity. Implementation of modular concepts, building block design, platforms and interface design represent examples for challenges in the field of product complexity.

The complexity of products and product portfolios of companies is often directly linked to the existence of process complexity. For example, the increase of product functions and components can create the need for more development process steps. And those steps are interrelated and need to be coordinated. Consequently, the company's process flowchart can become more complex due to increasing product complexity. Further constraints like decreasing development time, international product portfolios or distributed development approaches can increase process complexity even more.

Organizational complexity is also interlinked with the complexity types mentioned above. Managing complex products and product portfolios and executing complex processes requires adequate organizations. "Conway's law" [12] described this fact almost 50 years ago: He stated that when organizations design systems, those designs are similar to the organization's communication structures. This statement links product structures with organizational structures.

With the knowledge about Conway's law it is interesting to investigate the appearance of structures in organizations and products. Whereas functional and matrix-oriented structures have become popular over the last decades, hierarchical structures are still common in many organizations. On the other side, many products became highly networked structures that do not follow a hierarchical approach. Therefore organizational design is an ongoing challenge in modern product development.

It is important to consider the interrelation between the four aforementioned complexity classifications for determining the kind of a specific complex challenge. The origin of complexity and its appearance or perception are not necessarily determined in the same field. For example, an enterprise can possess a significant amount of complexity based on its comprehensive product portfolio. However, the originating cause of this complexity can sometimes be found in the markets the enterprise delivers to. It is important to identify the origin of the complexity in order to assess its value. That means a complex product portfolio does not possess any value in itself, but it can be the reason for high amounts of effort for the enterprise. In the context of an existent market complexity however, product complexity can be valuable as the product portfolio allows the delivery of the right products to this broad and diversified market.

In all four fields described above complexity results when a large number of system elements are mutually interlinked. That means it is not the pure number of product components, process steps or organizational roles, but instead their mutual dependencies that decide their complex behavior. Such systems become non-transparent for people interacting with them. A good example for a non-complex system with a huge amount of elements could be the database storing an enterprise's customers: even if many addresses might be included, this system only contains a few and obvious interdependencies (e.g. ZIP codes and cities). And if one address gets updated, deleted or added this does not have any impact to other system elements. In contrast to that, the database storing the requirements for a technical product of the same enterprise can be complex, as requirements possess many mutual interdependencies. Consequently, changes to one requirement can result in tremendous impact to many others.

In the field of product design and development, a complexity definition according to cybernetics is helpful [13]. In cybernetics simple, complicated and complex problems are distinguished. In contrast to simple problems, complicated problems are characterized by many highly interconnected parameters. Complex problems, in addition, possess high dynamics within the system. Ashby highlights that analysis methods designed for dealing with simple systems do not work for complex systems [7]. This can also be said for applying methods to complicated and complex systems, as it has been illustrated in the previous Sect. 3.1. Obviously, rules for classifying systems as simple, complicated or complex are not as explicit as pure mathematical complexity definitions. But for systems which cannot be algorithmically modeled, this represents a useful initial guideline.

Musès also categorizes complexity into three groups and from a practical perspective. Complexity I is inherited and exists in almost every system. Consequentially, it is difficult or even impossible to avoid it. Complexity II is caused by wrong handling and the usage of

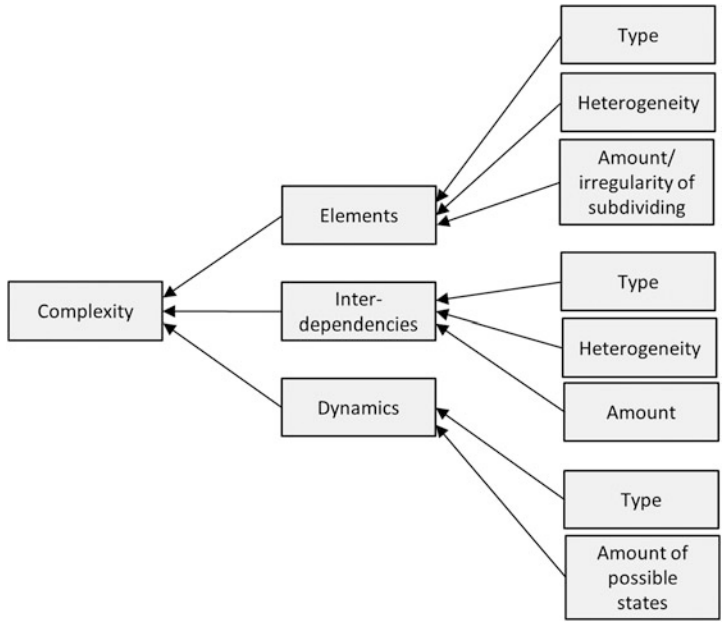


Fig. 3.7 Parameters of complexity from a product development perspective

incorrect approaches and therefore can be avoided, when better suited methods and procedures can be found and successfully applied. In contrast to that, Complexity III cannot be addressed with presently existing solutions. This type of complexity requires the creation of new and innovative methods to become manageable [14].

Lindemann defines complexity from a product development point of view [15]. He mentions the relevant parameters of complex systems to be the number of elements, their interconnections and resulting interfaces. In addition, he mentions that associated processes (e.g. design, production or distribution processes) contribute significantly to the resulting degree of complexity. Finally, Lindemann highlights that it is often important to link and integrate stakeholders into the system, which means to consider sociotechnical and not only technical system complexity. This reflects the initial thoughts of cybernetics' development, when Wiener saw the necessity to integrate operators of airplanes and air raid defense into the system modeling. In summary, Lindemann declares complexity to be dependent on the elements (type and heterogeneity, amount and irregularity of subdivision), the interdependencies (type, heterogeneity and amount) and the dynamics (type and space of possible states) [15]. This classification is shown in Fig. 3.7.

Lindemann mentions systems engineering as an approach towards complexity management [15]. A basic definition of complexity in this field has been presented by Sheard and Mostashari and is depicted in Fig. 3.8. Aspects like the size of a system, connectivity and architecture are similar to Lindemann's definition of complexity; additionally, in the systems engineering definition the aspect of environmental complexity is modeled

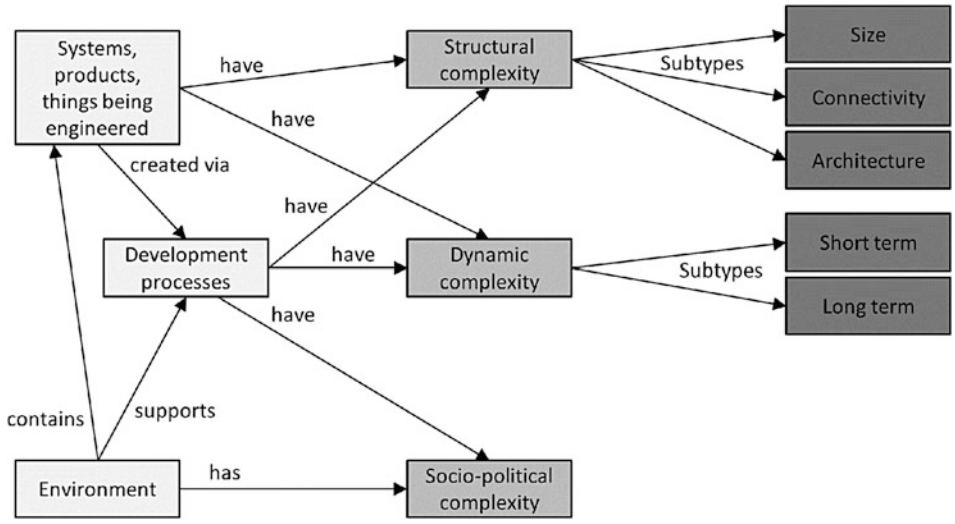


Fig. 3.8 Classification of complexity from a systems engineering perspective (adapted from [16])

explicitly. Three basic complexity types are defined—structural, dynamic and social-political complexity. While technical systems can have structural and dynamic complexity, socio-political complexity results from development processes and the environment.

In many engineering applications it can be useful to not only classify complexity by its type but by its origin. From an enterprise's point of view the separation of internal and external complexity can help in solving relevant challenges. Schuh and Schwenk introduced this perspective, depicted in Fig. 3.9 [18]. Here, internal complexity is understood similarly to Lindemann's perspective. This internal complexity emerges from the number of elements, interconnections and resulting interfaces. In addition, internal complexity can comprise process and organizational complexity resulting from an enterprise's effort of developing, maintaining and offering products or product portfolios. Thus, internal complexity results from a company's market offer.

External complexity emerges from the market requirements, e.g. the number and combination of functions requested by customers. This market-induced complexity can hardly be influenced by an enterprise and therefore represents an external source of complexity. Variant management is the challenge of matching the complexity of external market requirements with the complexity of the internal product offer. While the external complexity should be as large as possible (which means to fulfill a large variety of customer requirements), this needs to be realized with as little internal complexity as possible (which means to keep the internal efforts low) [18].

External complexity in the scenario of variant management contains a specific characteristic worth mentioning: this is one type of complexity that shall be increased, while in many other cases the objective is to decrease complexity. This external complexity can be

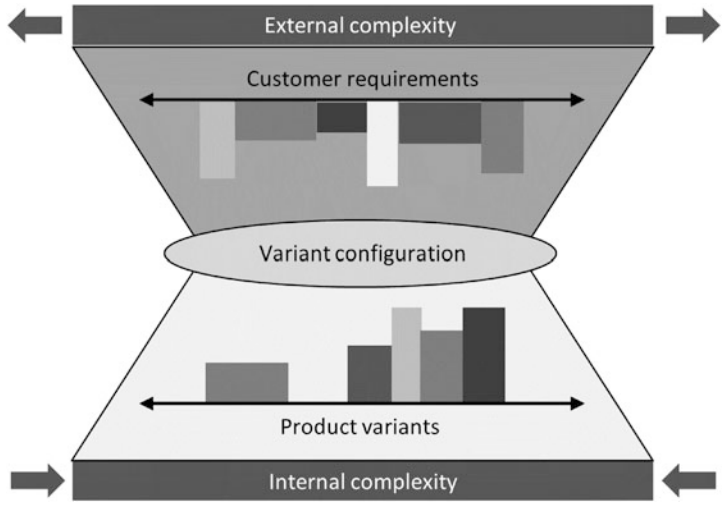


Fig. 3.9 The challenge of variant management at the interface of internal and external complexity (adapted from [17])

characterized as useful, compared to other useless types of complexity. The classification of useful and useless complexity is explained in detail in the context of a complexity management framework in Sect. 6.2.

3.3 Impact of Complexity

One basic part of a complex system is the large number of elements (variables) which are interconnected. While large is a vague term it is not possible to provide an exact number of elements that make a system complex. For example, scientists working with "systems of systems" like a city's infrastructure system would put the threshold of element numbers very high. But also systems with smaller numbers of elements can be in line with definitions of complexity.

Sheard summarizes complexity as follows: "Complexity is the inability to predict the behavior of a system due to large numbers of constituent parts within the system and dense relationships among them" [19]. That means that it is not a distinct threshold of system elements and interconnections that makes a system complex, but is instead the impact to people that results in a lack of understanding.

An impressive example is given by Browning in the context of Design Structure Matrices. These matrices represent matrix-based notations of elements and their interdependencies in a compact format. Browning mentions that even a number as low as ten elements can be difficult to oversee and manage [20]. The reason is that interdependencies between even a few elements can create high numbers of paths and

loops in the system. These can cause unexpected impact (side effects), because dependency chains become long, mutually overlap or build feedback loops. Long dependency chains are hard to identify, and overlapping dependency chains and feedback loops can aggregate to intensify impact or extinguish an effect. Especially feedback loops can cause unstable system behavior and the resulting effects can hardly be evaluated without high computational effort. System dynamics is an approach specifically dealing with feedback loops, which will be introduced in Sect. 3.4.3 in the context of engineering application and in Sect. 4.3.5 from a historic perspective.

Dynamics is another important characteristic of complex systems, which is mentioned in all different definitions of the term. However, the precise specification given from mathematics (see Sect. 3.2 and Fig. 3.5) is not helpful for application to engineering challenges, which cannot be fully algorithmically modeled. Nevertheless, the impact of dynamics to complex systems can be described. Norbert Wiener, the pioneer of cybernetics, was the first to model human operators and technical devices in an integrated model, which was helpful for solving the associated challenge; but such systems turned out to be complex control problems to solve. The development of cybernetics is described in Sect. 4.3.2.

As long as no interaction happens either between the environment and the system or between elements within the system, even a high interconnectivity between multitudes of system elements does not result in effects of complexity. Interaction with the system means that information is transferred via an interconnection and this action can initiate further interactions along connected elements and dependencies. If, however, no interaction happens, the interdependencies are inactive and therefore irrelevant. In other words, effects of complexity are associated with the application of a system.

An example can highlight the significance of this statement: Most people would agree that today's smartphones represent complex systems. People think so, as they might think about the many (interconnected) electronic components or the many software applications. But if one were to use a smartphone to participate in a mobile phone throwing competition (that really exists: http://www.mobilephonethrowing.fi/), no complexity is associated with the phone. In fact, it would make no difference if one uses the phone or a brick (of same size and weight) for the competition. The absence of complexity in this (rather unusual) use case results from the fact that the application does not trigger any informational impact to the technical system. And consequentially the system elements and interdependencies are irrelevant for this case. If, however, a developer has to apply a technical update to a smartphone, the effects of complexity can easily occur. The technical measure causes impact to the system, which can spread via interdependencies to many other parts of the system. In the worst case unpredicted effects can occur.

It needs to be mentioned that complex systems do not need external input for a dynamic behavior to initiate. Dynamics can emerge in the system itself and either the specification of elements/variables can change or their interdependencies. A typical example for an internal source is a failure of single element, which can result in tremendous consequences. These consequences mean that the system produces visible output that passes the system

boundary. Thus, the system is open. Closed systems do not interact with the environment; by definition they are isolated from it and every impact remains within the system. Bertalanffy classified systems according to their interaction with the environment [21]. His General Systems Theory is more closely explained in Sect. 4.2.

The large number of elements and interdependencies make a complex system non-transparent and incomprehensible to people, who can develop a fear of interacting with such a system [3, 22]. Obviously, this can significantly impact decision processes and lead to failures when dealing with such systems [3]. Dörner mentions four causes for failures when interacting with complex systems: Slowness of thinking, protection of one's own competence, minimal recording of information and fixation of attention to the actual problem only. The resulting failures can, for example, be observed and experienced in business games like the "Beer game" (see Sect. 4.3.5)—which illustrates immediate action, ineligible system or process reduction/abstraction or neglecting side effects resulting from insufficient problem understanding. Other failures have a psychological basis, e.g. endless planning without acting, solving known problems or ad hoc reactions.

Considering the possible impacts of complexity like the inability to make decisions or wrong decision-making, this points out its tremendous relevance. While uncontrolled complexity poses high risks to organizations, societies and enterprises, successful management of complexity also implies significant opportunities. For an enterprise context, Maurer mentions a lack of decision-making ability, frequent development crises and product changes as well as long development process duration as consequences resulting from complexity [23]. In addition, he describes that the effective managing of complexity can provide beneficial opportunities like increased competitiveness, successful control of large variant and product spectra and possibilities of increased product customization. And as managing complex systems is more challenging than dealing with simple systems, successfully managed complex systems imply a significant hurdle for copycat products and competitors entering the market.

Vester describes the risks of complexity for human societies. He states that due to an increasingly complex world (mentioning unemployment, dramatic environmental changes, stock market crashes and military conflicts), even well-planned interventions can lead to fatal consequences because of feedback loops and time delays [24].

Complexity is an integral part of many systems and seems to be required for realizing higher states of development. Complexity can be found in biological systems, and impressive functionalities such as that which is delivered by the human brain seem to be not achievable with simple system design [22, 25]. Thus, complexity can be naturally required for a system to work, and an indiscriminate strategy of complexity avoidance can be harmful. This does not imply that all complexity is necessary and helpful. In fact, it is important to distinguish useful and useless complexity and treat both kinds accordingly. This will be further explained in Sect. 6.2.

In 2012 the study "Mastering Complexity" from Camelot Management Consults tackled the relevance of complexity for the economy [26]. Eighty-three percent of the top managers surveyed (more than 150 participated in the study) mentioned that the degree of complexity in their enterprises was too high. Eighty-nine percent said that complexity increased within

the last 3 years and 76% said they expect complexity to further increase in the near future. The study mentions that enterprises could raise their EBIT by 3–5% if complexity management is successfully implemented. While the Camelot survey included top managers from a variety of fields, the industrial sector alone has seen an increasing relevance of complexity and its management for the future. Fifty-nine percent of the managers expect increasing sales figures for offers of complete system solutions.

3.4 Established Complexity Management in Engineering

Offering a large product portfolio to the market often implies complexity. Therefore, it is interesting to have a closer look at one of the most famous companies with an almost inconceivably large spectrum of products—Amazon.com. The company started out as an online book store, and then became the largest retailer in the USA. If one only considers the number of products, the ability to economically offer such a large product portfolio would seem to be highly complex.

While the success of Amazon is highly impressive, the product portfolio (considered as a system) does not possess many interdependencies. Changes to one product do not influence other products, the business processes or the organization. And new product requests from customers (the market) can be served with additional product offers that do not impact the enterprise's processes or organization.

The company Spreadshirt became popular by their offer of customized shirts with short delivery time. This business of minimal order size can be realized, because product development (creating the customized shirt) is not burdened by significant interdependencies. The same business model with for example customized combustion engines would most likely fail, as each customization would impact many other parts of the product as well as system elements in the process (production, testing) and organizational field. The challenge of managing system interdependencies explains why product customization approaches so far focus on simple systems only.

Mass customization, as a blend of the terms mass production and customization, has obtained much interest over the last 20 years. However, offered products like NikeID (www.nike.com/NIKEiD) or Reebok Custom (www.reebok.com/us/customize) only allow selecting from a predefined set of color options or decorations. Those customization tools represent configuration approaches and do not fulfill the mass customization idea of "producing goods and services to meet individual customers' needs with near mass production efficiency" [27]. Product customization approaches are applied on complicated but not complex systems.

For tackling complex problems in the engineering context, four main approaches have become established. These techniques are partly based on the same groundwork and mutual influence. The historic context will be explained in Chap. 4, indicating the steps from system awareness to modern complexity management. In the following sections these approaches shall be briefly introduced in terms of objective, functional scope and application context.

One fundamental approach of complexity management is not introduced in a separate section, as it comprises part of most other techniques—systematic visualization. System elements and their interdependencies contribute significantly to a system's complexity. If those system structures can be externalized, users can more easily understand and simulate system behavior. In engineering many visualization techniques get applied, and depending on the specific field some specifications have become quasi-standards, e.g. event-driven process chains or SysML models [28]. However, Sect. 3.4.4 introduces dependency modeling, which is based in large part on visualization techniques.

3.4.1 Operations Research

Operations research is often misunderstood as a collection of mathematical methods. While it is the application of many of these methods, it cannot be reduced to a purely computational approach. Quantitative models and methods are used to help find an optimal decision for complex challenges. Human decisions and non-quantifiable arguments are not considered in this part of the approach, but best integrated in the subsequent analytical part of the approach. Thus, operations research is not a straightforward solution-finding process, but provides support for optimal decision-finding.

Operations research is characterized by the cooperation of disciplines like mathematics, economics and computer science, and can be subdivided into sub-branches. Linear optimization, transport optimization, combined optimization and dynamic optimization are a few prominent ones. Also game theory often gets applied in operations research (see Sect. 4.3.6) [29, 30]. In general, deterministic and stochastic approaches are distinguished in operations research.

The main objective of operations research is to describe decision problems by optimization and simulation models, and to develop an algorithm that can be applied for solving the problem. Problems are classified by criteria like degree of information, (types of) target functions, and constraints and solvability. Therefore, decision, optimization and simulation models get applied [30]. Operations research focuses more on the system states than on the system structure and often applies experiments.

Based on the definition and objective of operations research, six phases of application exist. These phases do not represent isolated, straightforward process steps, but are highly interlocked with each other or can be worked on in parallel if required.

- Formulate the problem
- Develop a mathematical model
- Deduce a solution based on the model
- Validate the model and solution
- Supervise and adapt the solution
- Implement the solution

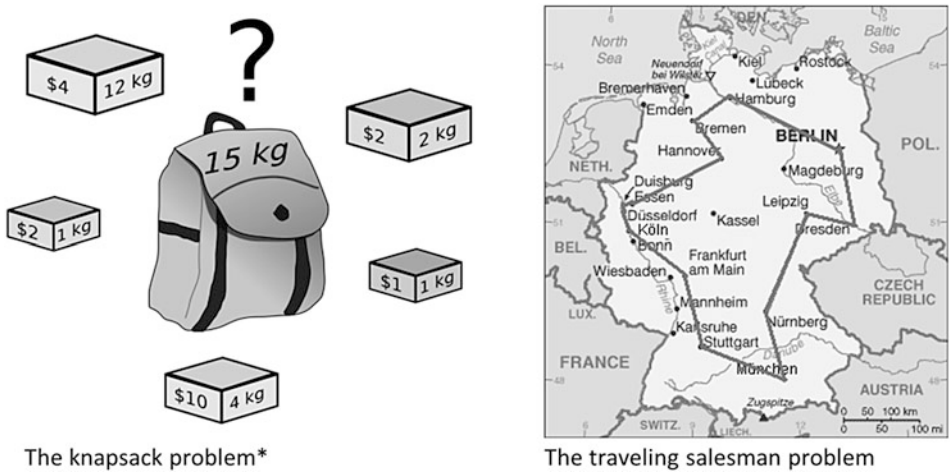


Fig. 3.10 Knapsack problem and traveling salesman problem, two complex challenges in operations research, *Dake, CC BY 2.5 (http://creativecommons.org/licenses/by/2.5), via Wikimedia Commons

A popular example for the application of operations research on a complex challenge is the knapsack problem [31]. The knapsack problem is based on the theoretical use case that one has to fill a backpack with objects of maximum value. The available objects have different sizes and values while the backpack provides a limited amount of storage space. While the example is hypothetical, this challenge appears in a large variety of real-world problems. For example loading trucks with payloads or organizing the workload of computing centers can be modeled by this abstract problem. The knapsack problem represents a challenge of computational complexity and can be mathematically described and solved—e.g. by dynamic programming approaches.

Another popular problem that can be solved by operations research methods is the traveling salesman problem. Here when given a set of predefined places, the target is to find the shortest path visiting each place exactly once and returning back to the starting point. Despite the fact that the situation and the constraints can be clearly described mathematically, it still represents a complex computational problem, as defined from a mathematical perspective (computable complexity, see Fig. 3.5). The problem can be solved using heuristic and approximation algorithms [32]. Figure 3.10 (right side) shows the shortest path to visit the 15 largest cities of Germany in a closed loop. This path represents one out of 43,589,145,600 possibilities.

3.4.2 Systems Engineering

Products and modern production facilities are often highly complex, large-scale projects. They contain an unmanageable combination of technical systems and impact factors: mechanics, electrics, energy, control systems, hardware and software, humans and

machines, logistics and communication, customers and suppliers. More and more product requirements are accompanied by increasingly faster development processes and strict constraints concerning budget, quality and time to market. With these conditions in mind, it becomes obvious that the development of for example a passenger airplane, an innovative luxury car or any other large system represents a huge task, which can only be solved by interdisciplinary teams. The required knowledge for such projects is extremely manifold, multidisciplinary and extensive. While it is impossible to concentrate this knowledge in only one or a few people, there is the necessity of project planning and decision-making, which requires well-informed people with a comprehensive project overview. The need for such system specialists increased with the beginning of the age of industrialization and the associated technological advances.

The systems engineering approach was developed to meet these upcoming needs for system specialists. Initial applications were conducted after the Second World War. The most famous use case became the application of systems engineering for Project Apollo, the US aerospace program in the early 1970s.

The original term systems engineering can be traced back to Bell Laboratories in the United States in 1940, where it became mentioned in the context of weapons system development. Related and partly integrated approaches and terms are e.g. systems architecture, systems engineering management and systems design. Those terms are partly used as synonyms, and make a unified definition of systems engineering rather challenging. The common denominator of all these approaches is that for solving a complex problem, the problem gets disassembled into smaller parts and reintegrated later into a final solution [33].

In German-speaking countries different translations of systems engineering are in use, which can be misleading. While the English term systems engineering is more commonly used (e.g. the German Chapter of INCOSE (International Council of Systems Engineering) is named “Gesellschaft für Systems Engineering”) [34], terms like Systemtechnik (German for system technology) and Technische Systemanalyse (German for technical system analysis) are also applied. While Technische Systemanalyse focuses on technical system design, Systemtechnik typically includes associated procedures like project management. Then terms like system management or system control are also applied [35]. Saynisch classifies systems into three categories: material or object systems, process-related systems and target systems. He sees systems engineering as planning and designing technical (object) systems and representing the developmental and creative part, which is complemented by project management as the controlling part [35]. This differentiation between systems engineering and project management represents a modern view, while older definitions saw project management being a building block within the systems engineering approach, e.g. [36].

Systems engineering can be applied to a technical product or a superior man-machine-system, including the application of a product. According to Bluma, systems engineering is the integration of different components into a technical system. The components are considered by their function for the entire system [37]. Bluma mentions that the system

is not constrained to the technical components only, but can also integrate non-technical elements of system organization and system use into the analysis. This allows treating each engineering problem with a systems engineering approach. This approach integrates widely fragmented sub-disciplines into one engineering approach and formed the concept of a systems engineer. Several methods and models of systems engineering originate from cybernetics, e.g. black-box principles, block diagrams and statistical system analyses [37]. The holistic approach of systems engineering is well-suited for being applied in solving complex problems. Systems engineering tools make use of qualitative as well as quantitative methods.

3.4.3 System Dynamics

Many questions of society, corporations and organization are complex challenges, as the system behavior is often nonlinear—cause and effects cannot be fully understood and system impacts can be unpredictable. For such challenges often the approach of system dynamics is applied. System dynamics is a mathematical approach that models complex systems using flows between system elements that can form feedback loops and stocks [38]. Developed in the 1950s by Jay Forrester, it became a powerful computer-supported modeling approach that is widely used for complex problems. The most famous application is the modeling of exponential growth for the world, described in the publication The Limits to Growth [39].

In system dynamics, four basic concepts can be differentiated: The core is one concept which describes information feedback, meaning that interactions between system components can be even more important than the components themselves. The system behavior is mainly resulting from feedback loops between its elements. Feedback loops can be classified into positive and negative feedback. They possess a specific structure and impact can occur with delay [40–42]. Automated decision-making is the second concept, which is based on a military approach of strategic, long-term decisions including their mathematical modeling [42]. Computer simulations enable a basic understanding of complex system behavior and represent the third concept of system dynamics. For example, different management approaches or market expectations can be assumed and their potential consequences can be estimated by computer simulations [38, 42]. The fourth concept then is the use of digital computers, which is required for conducting simulations and representations of complex systems [40].

An important part of system dynamics is the modeling of a complex, dynamic system. Forrester proposed a six-stage modeling process, guiding the user from the problem to a solution. Forrester, as well as his former student Barry Richmond, were both focused on quantitative modeling approaches, as they thought it was the only way to understand dynamic system behavior. Geoff Coyle was another student of Forrester and later founded the system dynamics group at the University of Bradford. Coyle had a contrary opinion concerning modeling approaches, which led to a long-lasting scientific dispute [43, 44].

The main aspects and differences between quantitative and qualitative approaches in system dynamics modeling shall be briefly explained. First of all, the type of modeling depends on the selected representation form, which can be distinguished into four categories: verbal descriptions, cause-and-effect diagrams, flow diagrams and mathematical equations [45].

According to Ossimitz, verbal descriptions of system models are easy to understand but rather unprecise. Consequentially, these descriptions can only usefully be applied to qualitative system model descriptions [45].

Cause-and-effect diagrams are typically composed of graphs built up from nodes and edges. Here, system elements are represented by nodes, while the impact from one system element to another gets modeled by edges connecting two nodes. Edges can be specified by plus or minus symbols to provide a more detailed description of the cause-and-effect relationship between the two nodes. A plus symbol indicates a monotonically increasing relationship, while a minus symbol indicates a monotonically decreasing relationship. If all edges that form a closed feedback loop are specified by plus or minus symbols, then a restraining or escalating behavior can be identified. Ossimitz explains that it is not required to specify system elements in a cause-and-effects diagram with numbers. However, some system elements can already imply a quantitative specification. Ossimitz mentions the example of car traffic, where either cars on the streets in general or a specific amount can be expressed [45].

The third form of representation is the flow diagram. This can be seen as an extension of the cause-and-effect diagram by different types of system elements and relations. System elements are separated into state/level/stock variables, flux/rate/flow variables and auxiliary variables. Standardized symbols are applied for representing system elements in flow diagrams [44, 46]. The meaning of this differentiation is that timed processes often require distinguishing between variables that possess a value at a certain point of time and variables that possess a value after a specific time. One example could be the balance sheet of an enterprise, which comprises state variables for a specifically selected day. In contrast to that, the profit and loss statement comprises flow variables related to the flow rate of 1 year. For reasons of easier simulation, continuous processes are also modeled with state and flow variables just like in the discrete case.

The fourth option of representation in system dynamics is the application of mathematical equations. These equations only comprise quantifiable variables and typically build the basis of a numerical simulation [45].

In system dynamics, qualitative as well as quantitative models can be applied. In practical applications, often the problem situation gets acquired and structured first by applying qualitative modeling. Quantification represents a follow-up step, which allows for simulation-ready models that can be used for scenario analyses [47]. Also Jay Forrester holds the opinion that qualitative approaches should be supplemented by acquiring a subsequent quantitative system dynamics model. He argued that qualitative methods do not meet the requirements of dynamic simulations [43, 44].

3.4.4 Dependency Modeling

Dependency modeling does not represent a separate discipline, approach or technique. Here the term is used as an aggregation for those methods aiming to support the management of structural complexity. When these methods were developed, visualization of system structures was one of the main objectives. With the increase in computational power and the big data approaches, visualization became more abstract and computational methods replaced manual system interaction. Nevertheless, structural modeling and its associated methods of complexity management are still widely in use.

Already with the rise of cybernetics, graphs were applied for indicating system dependencies. However, matrix representations were more common because they were already integrated in applied mathematics. Graph visualizations can be very intuitive, and many dependency models make use of them. Especially process models often get visualized with graphs, and some became quasi-standards for whole industry branches, for example event-driven process chains. Browning and Ramasesh provide a comprehensive overview of applied process models in product development [48]. Graph depictions of dependency networks typically require much drawing space and can become more confusing the large the networks become. This is mainly because of crossing and long-range dependencies, which make it difficult to understand and interpret the network.

Matrix models of dependency networks make it easier to apply computational methods, which is one of the reasons for their intensive use. In addition, matrices provide the basis for systematic dependency acquisition processes. Alexander was the first to document the acquisition of dependencies in a matrix with a symmetric arrangement of elements on both axes, which created a pattern for depicting interdependencies in the matrix cells [49]. For this specific matrix form, techniques of reordering elements and dependencies have been developed, which allows one to analyze and optimize dependency networks. These square matrices and the methods of their manipulation get aggregated under the term Design Structure Matrix (DSM) and have been developed since the early 1960s [50]. Methods have been further developed since then and many industry applications are documented [51].

One of the main applications for DSM is the identification of useful modules in a system by rearranging matrix columns and rows, see e.g. [23, 52]. As well, process or task sequences can be streamlined using the same technique. Yassine and Braha mention, “The DSM approach to managing complex development projects is an information exchange model which allows the project or engineering manager to represent important task relationships in order to determine a sensible sequence for the tasks being modeled” [53]. A comprehensive introduction to DSM methods is provided by Eppinger and Browning as well as Lindemann et al. [8, 51].

Figure 3.11 shows a typical DSM with elements listed in the row and column headings, and interdependencies between the elements indicated by dots in the inner matrix cells. The location of dependencies shows potential for module-building, as most dependencies are

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

Fig. 3.11 Design Structure Matrix (DSM), adapted from [54]

clustered in a block. However, in terms of finding an optimal module structure the elements need to be further relocated in the matrix order.

DSMs represent a highly compact form of dependency modeling. They are definitely not as intuitive as a graph representation; however more elements can be depicted with DSMs. And once familiarized with the methods, one can visually identify specific structural patterns, e.g. clusters, straightforward dependency chains, feedback loops or isolated system areas.

The DSM is complemented by the Dependency Mapping Matrix (DMM) representing a matrix with two different types of elements on either axis [55]. This format allows representing dependencies between two types of elements, while the DSM documented dependencies of elements within one type. The format of a DMM is very common in many other applications and also outside of the field of engineering. Examples of applied names are “cause and effect matrix” or “interface structure matrix” [56, 57]. However, the DMM approach also comprises analysis and optimization methods for system dependency networks. For example the clustering of a structure represented in a DMM allows for simplifying the dependency management between two types of elements in a system [58–60].

Figure 3.12 on the left side shows the modeling capabilities of a DSM in graph format, which is limited to one element type and one dependency type only. The DMM enlarges the possibilities of system modeling to two types of elements, while still only one dependency type can be depicted (illustrated on the right side of Fig. 3.12).

Looking at Fig. 3.12 it becomes obvious that DSM and DMM can only model small system parts, e.g. the internal component structure or associations of people to tasks or

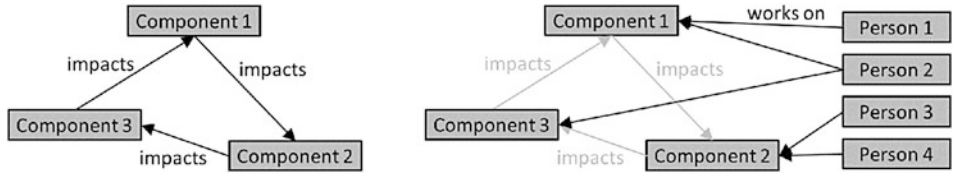


Fig. 3.12 DSM and DMM representation capabilities in graph format

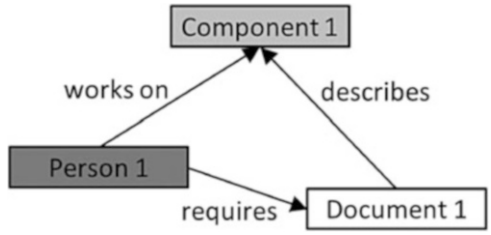


Fig. 3.13 Several element types and dependency types form the reality in many systems

components. However, system complexity is very much about the holistic perspective including various element types. One of the central claims of cybernetics was the interdisciplinary approach, which naturally includes many element and dependency types. Figure 3.13 illustrates a very simple system composed of three elements belonging to three different element types. And between them three dependencies of different meanings exist. Such a system could not be represented nor analyzed with the means of DSM or DMM. These approaches only extract single system views for further investigations, e.g. the component network.

A more holistic approach on dependency depiction is provided by the Multiple-Domain Matrix (MDM), which allows modeling several element and dependency types and deriving specific system views for closer analysis and optimization [4]. An abstract illustration of an MDM is shown in Fig 3.14. This matrix shows the element types (domains) and not the single elements. Each of the five domains (components, people, data, processes and milestones) contains a distinct number of elements. The inner fields of the matrix in Fig. 3.14 represent sub-matrices containing the dependencies between the elements of the types listed in the column and row headings. The words in the abstract matrix fields indicate the meaning of the dependencies.

The upper left matrix field (indicated by 1) contains change dependencies between components. Thus, this matrix represents a DSM, as do the four other matrices along the diagonal of the MDM. The next matrix to the right (indicated by 2) contains dependencies between components and people; here a dependency means that a specific component is "processed by" a specific person. Because of the two different element types on the two axes, the matrix is a DMM, and so are all the off-diagonal matrices in an MDM.

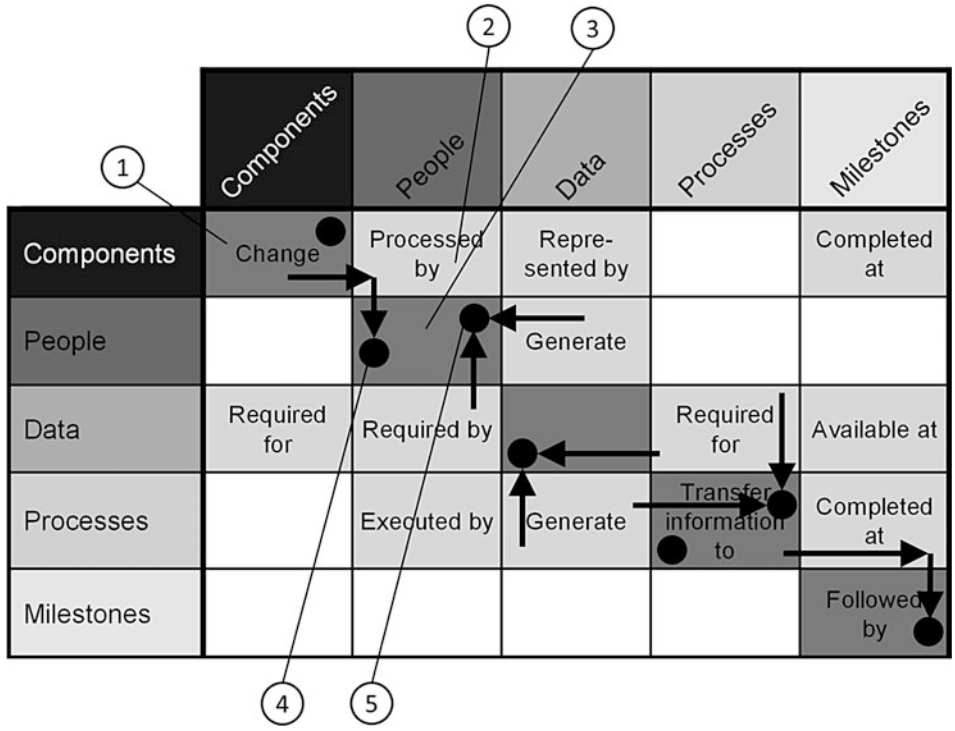


Fig. 3.14 Multiple Domain Matrix (MDM)—holistic system structure analysis

The matrix field indicated by 3 is a DSM field for the element type "people". The fact that no dependency meaning is noted in the field shows that no dependencies have been acquired and so far the matrix field is empty. However, dependencies between people can be computed based on other dependencies in the MDM. As examples, these possibilities are indicated by the black arrows leading into the matrix field. Such matrix computations result in indirect dependencies, which aggregate dependency information from two (or even more) element types into one system view.

The first computation uses the DSM of component dependencies and the DMM between components and people, and is indicated by 4. The resulting (indirect) dependencies between people indicate that two specific people are connected because they work on components that are linked by a change impact. If the modeled people represent product developers working on parts of a technical system, this system perspective shows their mutual dependencies based on their component responsibility. Interpreting this network could be used, for example, to optimize the organization of group meetings. Figure 3.15 shows an exemplary network of people that can result from the matrix computation.

Another computation for creating a people DSM is indicated with the second chain of arrows and the indicator 5. In this case, the DSM is computed from the information "people who generate data" and "data required by people". While both networks are DMMs, they

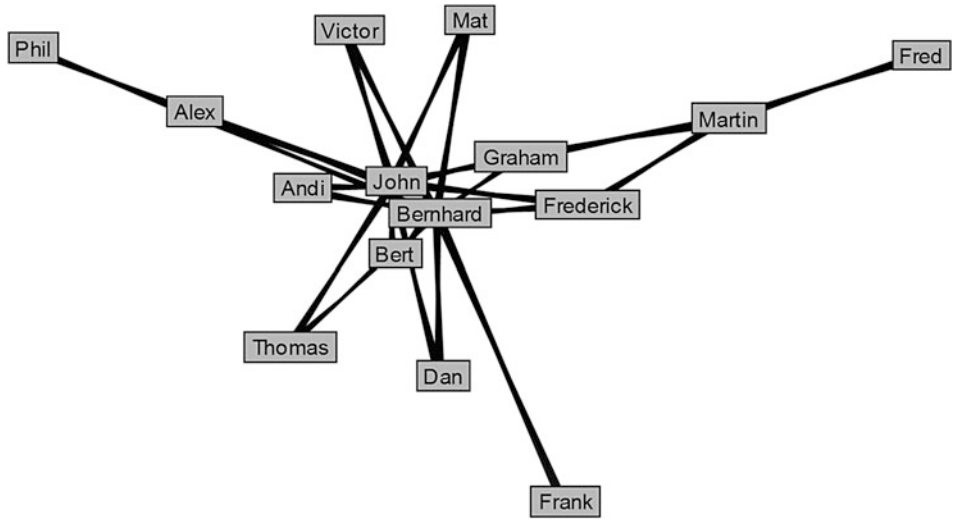


Fig. 3.15 Designers' dependencies based on their responsibility for components [23]

have different meanings and consequentially different dependency networks. An example result of the people network is shown in Fig. 3.16. The meaning of this computed network is that people are interconnected because a person generates data that is required by another person. This network showing the provision and request of data between e.g. product developers could be applied in improving the team organization in case of staff fluctuation. The computed dependency network provides a significantly different perspective, with a different group of people in the core of the network than in the network before. This indicates the importance of defining and creating the correct system views for answering specific questions.

The short example shows the major benefits of an MDM, which are capturing different types of system elements in a comprehensive model as well as computing and selecting specific system perspectives. These perspectives represent DSMs or DMMs, which means that the well-established methods of analysis, optimization and visualization can be applied. Maurer provides a comprehensive introduction to MDMs [60]. Industrial use cases are described by Lindemann et al. as well as Eppinger and Browning [8, 51].

Dependency modeling represents an important step in many approaches towards complexity management. Information visualization might be one of the popular fields of application, as many impressive network depictions have been created. The website "visualcomplexity" shows hundreds of large-scale networks with many different graph visualization approaches [61]. For most of these projects, information acquisition has been conducted by automatic tracking or data mining. Thus, the effort for collecting and preparing a large amount of system elements and dependencies was manageable. If information has to be acquired from people by interview—e.g. for documenting the

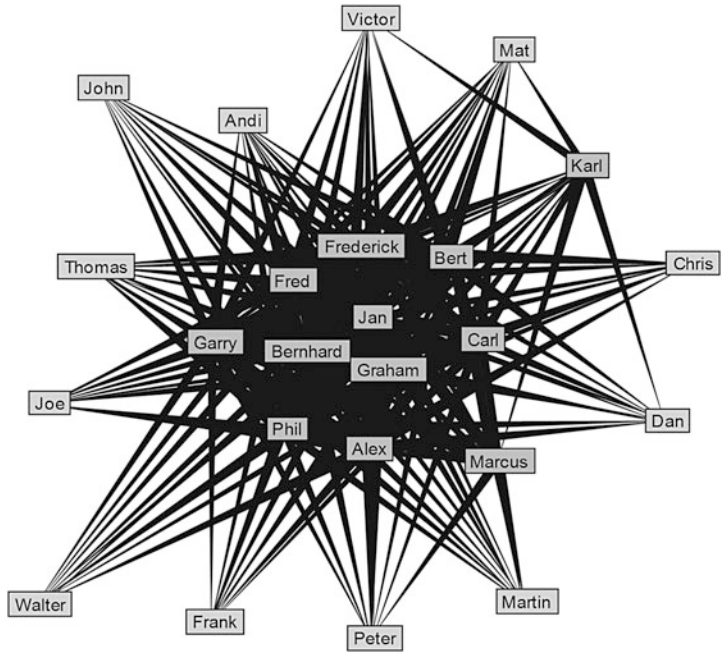


Fig. 3.16 Designers' dependencies based on their request and provision of data [23]

development processes, tasks and exchanged data between them—then dependency modeling also provides the approach of systematic data acquisition. More details on information acquisition can be seen in Sect. 6.4.

One characteristic of many complex challenges in engineering is the lack of transparency. This is especially because dependencies are often unclear or even unknown. And people involved in a project might have different understandings of the dependencies without knowing about the discrepancies. Dependency modeling makes system dependencies explicit and therefore stimulates discussions and a common understanding of situations and systems.

References

1. Mainzer, Klaus. 2008. *Komplexität*. Paderborn: Wilhelm Fink.
2. INCOSE. 2009. Systems Engineering Complexity Types.
3. Dörner, Dietrich. 1992. *Die Logik des Misslingens*. Reinbek: Rowohlt Taschenbuch.
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. Maier, Mark W. 1998. Architecting Principles for Systems-of-Systems. *Systems Engineering* 1 (4): 267–284. doi:10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.

6. Ehrlenspiel, Klaus. 2007. *Integrierte Produktentwicklung—Methoden Für Prozessorganisation, Produkterstellung Und Konstruktion*. 3rd ed. München: Hanser.
7. Ashby, Ross. 1961. *An Introduction to Cybernetics*. Fourth imp. London: Chapman & Hall.
8. 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.
9. Kolmogorow, A.N. 1965. Three Approaches to the Quantitative Definition of Information. *Problems Information Transmission* 1(1): 1–7.
10. Peak, David, and Michael Frame. 2013. Komplexität—Das Gezähmte Chaos. Springer.
11. Simon, Herbert. 1962. The Architecture of Complexity. *Proceedings of the American Philosophical Society* 106(6): 467–482 .http://nicoz.net/images/ArchitectureOfComplexity.HSimon1962.pdf
12. Conway, M. 1968. How Do Committees Invent? *Datamation* 14: 28–31.
13. Wiener, Norbert. 1948. *Cybernetics or Control and Communication in the Animal and the Machine*. New York: Technology Press.
14. Musès, C. 2002. Simplifying Complexity: The Greatest Present Challenge to Management and Government. *Kybernetes* 31: 962–988. doi:10.1108/03684920210436282.
15. Lindemann, Udo. 2005. *Methodische Entwicklung Technischer Produkte*. Berlin: Springer.
16. Sheard, Sarah A, and Ali Mostashari. 2010. A Complexity Typology for Systems Engineering.
17. Schwanitz, D. 1999. *Bildung. Alles, Was Man Wisen Muss*. Frankfurt: Eichborn.
18. Schuh, Günther, and Urs Schwenk. 2001. *Produktkomplexität Managen. Strategien, Methoden, Tools*. München: Hanser Fachbuch.
19. Sheard, Sarah. 2012. Assessing the Impact of Complexity Attributes on System Development Project Outcomes.
20. 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.
21. Bertalanffy, Ludwig v. 1973. *General System Theory: Foundations, Development, Applications*. 4th ed. New York: George Braziller.
22. Pruckner, M. 2002. 90 Jahre Heinz von Förster. Die Praktische Bedeutung Seiner Wichtigsten Arbeiten. Malik Management Zentrum St. Gallen.
23. Maurer, Maik S. 2012. Komplexitätsmanagement Für Die Industrielle Praxis—Komplexe Systeme Und Ihre Eigenschaften. Technical University of Munich.
24. Vester, Frederic. 2007. *The Art of Interconnected Thinking—Ideas and Tools for Tackling Complexity*. München: Mcb Verlag.
25. Holland, John H. 2000. *Emergence—From Chaos to Order*. Oxford: Oxford University Press.
26. Camelot Management Consults. 2012. Mastering Complexity.
27. Kaplan, Andreas, and Michael Haenlein. 2006. Toward a Parsimonious Definition of Traditional and Electronic Mass Customization. *Journal of Product Innovation Management* 23(2): 168–182. doi:10.1111/j.1540-5885.2006.00190.x.
28. Delligatti, Lenny. 2013. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. Boston, MA: Addison-Wesley Professional.
29. Beckmann, M., H. Gehring, K.-P. Kistner, C. Schneeweiß, G. Schwödiauer, H.-J. Zimmermann, and T. Gal, eds. 1992. Grundlagen des Operations Research 3: Spieltheorie, Dynamische Optimierung Lagerhaltung, Warteschlangentheorie Simulation, Unscharfe Entscheidungen. 3. Band. Springer.
30. Hillier, J.H., and G.J. Lieberman. 2004. *Introduction to Operations Research*. 8th ed. Boston, MA: McGraw-Hill Higher Education.
31. Kellerer, Hans, Ulrich Pferschy, and David Pisinger. 2004. *Knapsack Problems*. Berlin: Springer.

32. Applegate, D., R. Bixby, V. Chvátal, and W. Cook. 2006. *The Traveling Salesman Problem*. Princeton, NJ: Princeton University Press.
33. Weigel, Annalisa. 2000. An Overview of the Systems Engineering Knowledge Domain. MIT. http://web.mit.edu/esd.83/www/notebook/sysengkd.pdf.
34. Gesellschaft Für Systems Engineering E. V. 2015. www.gfse.de. Accessed 29 Dec.
35. Saynisch, Manfred. 2008. Systems Engineering Und Projektmanagement in Deutschland. Projektmanagement Aktuell 3: 1–9. www.pmaktuell.org/uploads/PMAktuell-200803/PM_3_08-025lang.pdf.
36. Daenzer, W.F., and F. Huber. 1999. *Systems Engineering: Methodik Und Praxis*. 10th ed. Zürich: Verl. Industrielle Organisation.
37. Bluma, L. 2004. *Norbert Wiener Und Die Entstehung Der Kybernetik Im Zweiten Weltkrieg*. Münster: LIT.
38. Forrester, Jay W. 1963. *Industrial Dynamics*. Cambridge, MA: MIT Press.
39. Meadows, D.H. 1972. *The Limits to Growth*. Reissue. Verlag Signet.
40. Hahn, F. 2006. *Von Unsinn Bis Untergang: Rezeption des Club of Rome und Grenzen des Wachstums in der Bundesrepublik der frühen 1970er Jahre*. Freiburg: Albert-Ludwigs-Universität.
41. Richardson, G.P. 1999. *Feedback Thought in Social Science and Systems Theory*. Waltham, MA: Pegasus Communications.
42. Tohum, M.H. 2008. System Dynamics—Betriebswirtschaftliche Anwendungsgebiete. Universität Koblenz-Landau.
43. Coyle, R.G. 2000. Quantitative and Qualitative Modeling in System Dynamics: Some Research Questions. *System Dynamic Review* 16(3): 225–244.
44. Kapmeier, F. 1999. Vom Systemischen Denken Zur Methode System Dynamics. Universität Stuttgart.
45. Ossimitz, G. 1991. Darstellungsformen in Der Systemdynamik. In *Anschauliche Und Experimentelle Mathematik*, ed. H. Kautschitsch, 175–184. Wien: HPT.
46. Coyle, R.G. 1996. *System Dynamics Modelling—A Practical Approach*. London: Chapman & Hall.
47. Gabler Wirtschaftslexikon. 2015. http://wirtschaftslexikon.gabler.de/Archiv/143837/system-dynamics-v5.html. Accessed 29 Dec.
48. Browning, Tyson R., and Ranga V. Ramasesh. 2007. A Survey of Activity Network-Based Process Models for Managing Product Development Projects. *Production and Operations Management* 16(2): 217–240. doi:10.1111/j.1937-5956.2007.tb00177.x.
49. Alexander, C. 1964. *Notes on the Synthesis of Form*. Cambridge: Harvard University Press.
50. Steward, Donald. 1981. The Design Structure System: A Method for Managing the Design of Complex Systems. *IEEE Transaction on Engineering Management* 28(3): 79–83.
51. Eppinger, Steven D., and Tyson R. Browning. 2012. *Design Structure Matrix Methods and Applications*. Cambridge, MA: MIT Press.
52. Fernandez, C.I.G. 1998. *Integration Analysis of Product Architecture to Support Effective Team Co-Location*. Cambridge, MA: Massachusetts Institute of Technology.
53. Yassine, Ali, and Dan Braha. 2003. Complex Concurrent Engineering and the Design Structure Matrix Method. *Concurrent Engineering* 11(3): 165–176. doi:10.1177/106329303034503.
54. Pimmler, Thomas U., and Steven D. Eppinger. 1994. Integration Analysis of Product Decompositions, no. September.
55. Danilovic, Mike, and Tyson Browning. 2004. A Formal Approach for Domain Mapping Matrices (DMM) to Complement Design Structure Matrices (DSM). In Proceedings of the 6th Design Structure Matrix (DSM) International Workshop. Cambridge, UK: University of Cambridge, Engineering Design Centre.

56. Allen, T.T. 2006. *Introduction to Engineering Statistics and Six Sigma—Statistical Quality Control and Design of Experiments and Systems*. London: Springer.
57. Kusiak, A., C.-Y. Tang, and Z. Song. 2006. Identification of Modules with an Interface Structure Matrix. ISL_04/2006. Iowa.
58. Danilovic, Mike, and H. Börjesson. 2001. Managing the Multiproject Environment. In Proceedings of the 3rd Dependence Structure Matrix (DSM) International Workshop. Cambridge: Massachusetts Institute of Technology.
59. ———. 2001. Participatory Dependence Structure Matrix Approach. In Proceedings of the 3rd Dependence Structure Matrix (DSM) International Workshop. Cambridge: Massachusetts Institute of Technology.
60. Maurer, Maik S. 2007. *Structural Awareness in Complex Product Design. Lehrstuhl Für Produktentwicklung*. München: Dr. Hut.
61. Lima, Manuel. 2011. *Visual Complexity—Mapping Patterns of Information*. New York: Princeton Architectural Press.