

2. The foundations of complexity metrics

To analyze the structure of engineering design processes using metrics, foundations from the different areas of relevance shown in Figure 2-1 are used. These were identified using the DRM (Design Research Methodology) approach [BLESSING & CHAKRABARTI 2009, p. 63].

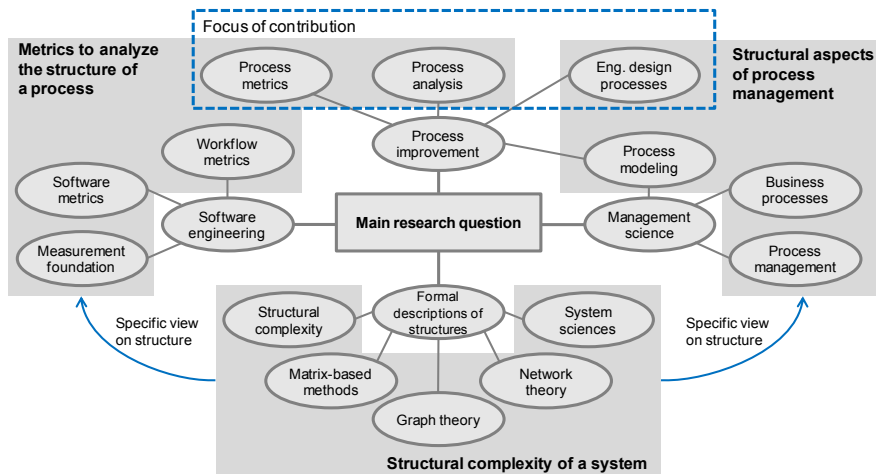


Figure 2-1: Main areas of relevance to this research

In this chapter, each area is explained in detail and related to the needs of this research. First, the foundations of systems in general and the structure are explained. Different means of dependency modeling for systems are explained, including relevant analysis methods. These foundations are used to explore what structural content is relevant for process management regarding both the goals of an analysis and the actual process models. Last, existing metrics are reviewed that are able to assess structures in general and, more specifically, in processes.

2.1 Structural complexity of a system

Complexity is present in many disciplines. Commonly, complexity means “consisting of parts or entities not simply coordinated, but some of them involved in various degrees of subordination; complicated, involved, intricate; not easily analyzed or disentangled” [SIMPSON et al. 1989]. Indeed, complexity has many facets. Computational complexity refers to the computability of an algorithm [PAPADIMITRIOU 1994]; information processing understands complexity as the total number of properties transmitted [NEWELL 1990]; and physics sees it as the probability of reaching a certain state vector [HEISENBERG 1999, HEISENBERG 2007]. In engineering, complexity generally addresses the high coupling of the entities of a technical system [MAURER 2007], and software science focuses on

assessing program code for its complexity, and thereby the risk of introducing errors into the code.

Complexity science originated from Cybernetics, founded by [WIENER 1948], and Systems Theory, founded for the most part by Ludwig von Bertalanffy [VON BERTALANFFY 1950]. It was also influenced by Dynamic System Theory, which belongs to the field of applied mathematics for the description of dynamic systems [PADULO & ARBIB 1974].

Structural Complexity Management is often seen as having evolved out of the first complex engineering projects that were accompanied by the paradigm of Systems Engineering, having itself evolved out of Systems Theory (e.g., the [NASA 1995]). The first use of Design Structure Matrices (DSM) is attributed to Don Steward [STEWART 1981], who used DSM to better plan complex projects involving many interdependencies, thus opening up the field of today's paradigm of structural complexity. Design Structure Matrices later evolved to Domain Mapping Matrices [DANILOVIC & BROWNING 2004], and then to Multiple-Domain Matrices [MAURER 2007]. While structural complexity generally regards technical (i.e., planned) systems, in parallel, Network Science describes complex systems of random or natural origin, such as the internet or molecules [BARABÁSI, 2003] [WATTS & STROGATZ 1998].

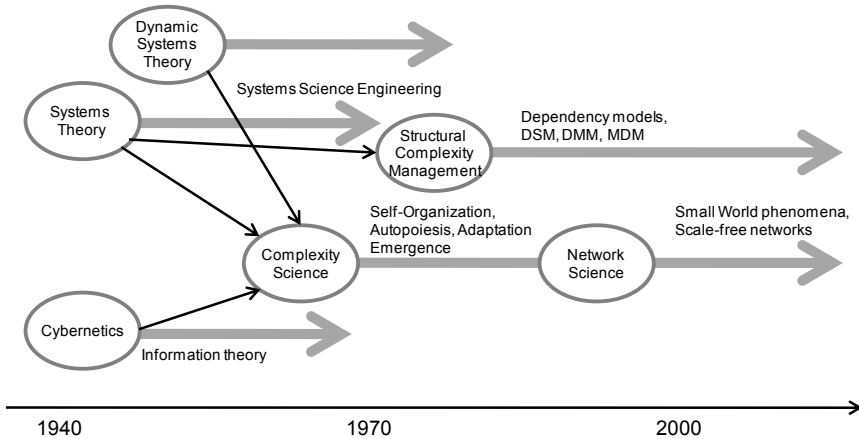


Figure 2-2: Evolution of sciences related to structural complexity (after [ART AND SCIENCE FACTORY 2009])

Figure 2-2 shows the development of each field of science from a historical point of view. The originating methods are discussed as follows. First, a review of systems and how they are constituted to derive a specific view on what the structure of a system is provided. For this purpose, the terms *system*, *complexity*, *structure*, *domain* and *relationship type* are explained. Using these basic notions, different kinds of dependency modeling are then introduced. Here, graph theory serves as a formal basis for all other dependency models and introduces a number

of concepts that keep appearing in other dependency models (e.g., cliques as tightly coupled clusters). Then, models and methods for matrix-based dependency models and for large networks are reviewed in detail to collect a complete set of analysis methods and the requirements to model the dependencies within a system.

2.1.1 General notions of managing structural complexity

Understanding processes as a complex system that has a network-like character and a certain structure necessitates certain basic notions and terminology, which are explained in this section.

System

The concept of a “system” is essential to the analysis of processes, as a process²⁵ represents a special form of a system [WASSON 2006]. A “system” is a set of entities that have relations among each other, either by interacting with or being independent of each other, and the system is delimited from the environment by a system border and connected to the environment by inputs and outputs [HALL 1963] [LINDEMANN 2007, p. 337].

WASSON defines a system as an “integrated set of interoperable entities, each with explicitly specified and bounded capabilities, working synergistically to perform value-added processing to enable a user to satisfy mission-oriented operational needs in a prescribed operating environment with a specified outcome and probability of success” [WASSON 2006, p. 18]. The “integrated set” refers to the fact that a system consists “of hierarchical levels of physical entities, entities, or components”. The entities are interoperable, as they must be compatible with each other to allow greater value of the system [BOARDMAN & SAUSER 2006]. This is also addressed by the “explicitly specified and bounded capabilities”, meaning that every entity “should work to accomplish some higher level goal”. To work “synergistically” underlines the additional value [RECHTIN 1991, p. 29] and the leverage factor of a network. Ultimately, the “value-added processing” alludes to the classical process term, which is based on the assumption that a process provides some additional value to the customer. This aspect of customer orientation is further emphasized by the expressions “specified outcome” and “success” in the definition [PATZAK 1982].

In a more general manner, MAURER provides a definition of a complex system that is, in many respects, similar to that of WASSON: “A system is created by compatible and interrelated parts that form a system structure, possess individual properties, and contribute to fulfill the system’s purpose. Systems are delimited by a system border and connected to their surroundings by inputs and outputs. Changes to parts of a system can be characterized by dynamic effects and result in a specific system behavior”. MAURER thus adopts the aspect of the relationships (“compatible and interrelated”) and boundaries (“system border”) but also provides a new aspect, that is, a system response that is not necessarily linear (“dynamic...specific system behavior”) to the definition [AUE & DUSCHL 1982] [MAURER 2007, p. 31].

²⁵ The term process will be defined later in section 2.2.1.

Keeping the focus of this research in mind, the following definition of a system is used:

A system is a set of entities of (possibly) different types that are related to each other via various kinds of relations. The system is delimited by a system border, across which inputs and outputs of the system are possible as an interaction with the environment. The system fulfills a purpose, which guides the meaningful arrangement of entities and relations. The behavior of the system is, in turn, due to the arrangement of the system's elements.

To a certain extent understanding a system denies traditional cause-and-effect chains and the certainty of determinism that, as practice shows, are not applicable to complex systems and complex processes anyway. This, however, implies that a reductionist approach which only centers on certain objects is too limited when regarding complex systems [BANATHY 1997].

Modeling complex systems

To manage a complex system, modeling is used to better understand it [BROWNING 2002]. In comparison to the object being modeled, a model²⁶ represents a target-oriented, simplified formation analogous to the original, which permits drawing conclusions based on the original [LINDEMANN 2007]. These conclusions normally comprise “making predictions and testing hypotheses about the effects of certain actions” [BROWNING 2002]. In this way, systems are made more transparent to improve understanding.

A model thus represents an abstraction of a real system, serving a certain purpose the model was made for; at the same time, a model usually involves pragmatics for those points that appear to be of little relevance to the purpose during the generation of the model [RECKER 2007]. This implies that the modeler influences the model in its expressiveness even before it is analyzed [MENDLING 2008, p. 7]. The development of any model-based analysis framework must thus incorporate possibly “unclean” models.

Nodes and edges, entities and relationships

In regards to a system's entities and relations, the management of structural complexity can be understood as an application of Graph Theory²⁷. As different terms prevail in the different disciplines, [Table 2-1](#) provides an overview of how the entities and relations within a system are commonly denominated.

²⁶ A good model acknowledges the “Guidelines of Modeling” to achieve inter-subjectivity of the models: correctness (i.e., the model is syntactically correct), relevance (only the interesting parts of the object being modeled are represented in the model), economic efficiency (the trade-off between the expense to build the model and making it as complete as possible), clarity (to ensure that a reader is able to understand the model), comparability (the consistent use of naming conventions and modeling schemes), and systematic design (the clear separation of different views) [BECKER et al. 1995].

²⁷ Graph theory is outlined in more detail in section 2.1.2.

Table 2-1: Different terminologies to describe the parts of a system

Term in Systems Theory	Entity	Relation
Term in Graph Theory	Node, vertex	Edge, arc
Other terms	Element: especially in matrix-based methodologies, the term “element” is used to refer to an entity that is entered into a row or a column	Relation: refers to any kind of association between two entities (directed or not) Dependency: often implies a direction

Domains

From a structural point of view, a system can be disentangled into a network-like model of entities and their relations. These entities can be of different kinds, e.g., documents, organizational units, and work packages. However, if many such kinds are mixed, a productive analysis is impossible. Each kind of entity represents a specific view, called a “domain”²⁸. The purpose of a domain is to create “homogeneous networks” that allow elements to be compared during analysis [MAURER 2007, pp. 71-72]. The term “domain” can, therefore, be defined as a specific view of a complex system, comprising one type of entity.

Three principles apply to the use of domains to model the structure of a complex system that involves specific views (visualized in):

- **Instantiation:** An entity is an instantiation of a domain. Thus, all entities that belong to one domain are of the same type, e.g., “information object”.
- **Decomposition:** A domain can be refined by creating sub-domains that are congruent with a part of the super-domain to which they belong, e.g., the domain “document” and the domain “prototype” can both be “information objects” in a process.
- **Recombination:** A system can be assembled by combining relevant domains and relationship types (see next page). By assembling different domains and relationship types, different views of the system can be generated.

Relationship types

Like the domains (see above), the relationships within a domain (or between two domains) need to be uniform to allow a systematic modeling and a purposeful analysis. While a domain contains entities of a kind, one relationship type refers to one class of relations that are similar [MAURER 2007, pp. 71-72]. The relationship type therefore defines the kind of (directed or undirected) relation between two entities. It is described as “(entity of domain A) is related to (entity of domain B)”. The term “related” can be replaced to specify the type of relationship; of course,

²⁸ A domain is comparable to a “class” of objects in object-oriented programming paradigms.

reflexive relationship types (“intra-domain”) are possible as well as mappings between two domains (“inter-domain”).

Again, three principles apply:

- **Instantiation:** A relation is an instantiation of a relationship type.
- **Differentiation:** A relationship type can be refined by detailing the description in a coherent manner with the superior relationship type. However, such a refined relationship type applies to fewer entities and is harder to model coherently [MAURER 2007, pp. 71].
- **Recombination:** A system can be assembled by combining relationship types.

Figure 2-3 visualizes these three principles for both domains and relationship types. At the highest domain it uses domains and relationship types to set up the meta-model of a system. Here, the basic class of an entity is defined. If a pre-defined meta-model is used, it actually provides the domains and relationships. If necessary, both domains and relationships can be refined and broken down into more specific sub-domains or differentiated relationship types. This is necessary, for example, when two domains are linked by two different relationship types simultaneously such that each transport a specific meaning, as shown in the example on page 131. To build a model, the chosen domains and relationship types are instantiated.

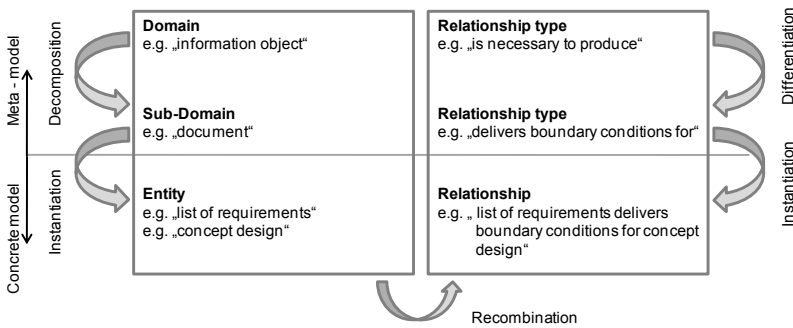


Figure 2-3: Principles of using domains and relationship types to describe the structure of a complete system

Figure 2-4 embodies these principles into the basic meta-model that is used for structural modeling, using only domains and relationship types and their decomposition. It represents a common dependency model that is only able to represent entities and relations.

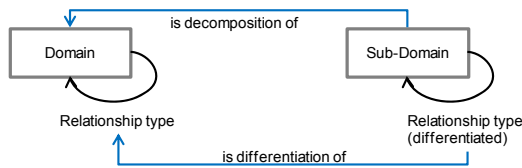


Figure 2-4: Meta-model for structural modeling

Views to a system: Native and aggregated networks

Complex systems can consist of more than one domain. These domains can be coupled within themselves or among themselves. In turn, it is possible that two domains are not directly related but only via a third domain. However, for many analyses it is necessary to use intra-domain networks, i.e., networks that consist of only one domain and of one relationship type that connects only the entities of the one domain among themselves. Therefore, aggregate views are needed at times to reduce the relationships via an intermediate domain to using just one single domain of reference.

In fact, according to the availability of models and data, a network can be *native*, i.e., the information in the model is a direct result of the data acquisition [MAURER 2007, p. 78] , or it can be *aggregated*, i.e., it is computed based on other networks that are available. Figure 2-5 visualizes the difference. In common Multiple-Domain Modeling (compare section 2.1.3), this aggregation is used to condense relations across one or more domains into single intra-domain networks to facilitate analysis. Section 2.1.3 illustrates in greater depth how this is done.

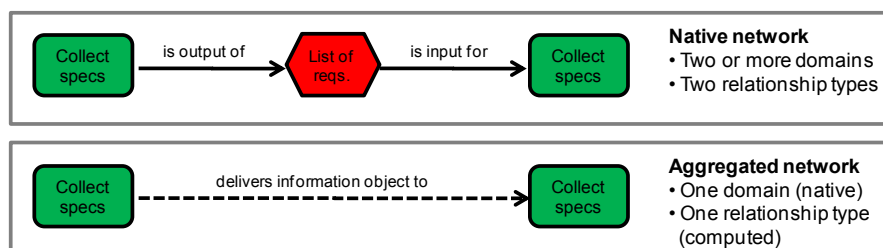


Figure 2-5: Native and aggregated networks

Structure

Although the term *structure* (Latin: *structura*: orderly combination, derived from the verb *struere*: combine, stack, stratify) appears to be one of the most common words in engineering terminology, there seem to be very few definitions available. Commonly, the term structure is used in engineering for organizational structures and product structures, i.e., the purposeful organization of an enterprise into

manageable entities or the setup of a product's components, functions, etc., into a meaningful architecture.

In mathematics (algebra and model theory), the term *structure* refers to a set of distinct entities, including the functions that transform these objects and the relations among the objects and functions. Systems Theory narrows this definition down to the patterns of interaction of the elements towards a behavior of the system [OLIVER et al. 1997, p. 21], which possibly serves a purpose [BOARDMAN & SAUSER 2006]. Structure is also closely connected to the phenomenon of emergence (see section 2.1.4) and sophistication, which is sometimes used as a measure for complexity and its logic decomposition into compact parts [KOPPEL 1987]. Both concepts indicate that structure involves the regrouping of entities of a system according to the logics (and the relationships between the entities) of a specific perspective. This is in line with the common understanding of structure as “the aggregate of entities in their relationships to each other” [MERRIAM-WEBSTER ONLINE DICTIONARY 2009]. A structure can be a hierarchy (a cascade of one-to-many relationships) or a network featuring many-to-many relationships [PULLAN & BHADSHIA 2000]. MALIK and BEER both refer to structure as the degree of organization which helps to bring order to a complex system and maintain it in order to enable the management of the system [MALIK 2003, p. 211] [BEER 1972, p. 65]. [GOLDENFELD & KADANOFF 1999] link complexity to structure (i.e., a basic pattern) by pointing out that everything has a basic structure according to how it is built; yet all systems come in variations that deviate from this basic structure, causing complexity. Structure is, therefore, defined as follows:

Structure regards the pattern in the network formed by dependencies (edges) between a system's entities (nodes). It reflects the semantics of this network; the structure of a system therefore always contributes – in its constellation – to the purpose of the system. A structure takes shape in structural characteristics, i.e. a particular constellation of nodes and edges. The characteristic gains its meaning by the way the pattern is related to the actual system it is part of, i.e. it must serve a special purpose in the context of the overall system. A structural characteristic only possesses significance in the context of the system it is describing.

In process management, structure refers to patterns within processes. Workflow patterns²⁹, as a particular case, represent the basic decision structures of a process [VAN DER AALST, et al. 2003]. These patterns, however, relate to possible constellations of splitting and joining the control-flow of a process using AND, OR, or XOR operators. Yet, the patterns relate to the different perspectives of a process, as also outlined by [CARDOSO, 2005a]: control-flow, data, and resources. This idea can be extended to detect possible errors in a process, i.e., they search process models for typical structures that point to an error in the model [VAN DONGEN et al. 2006] [MENDLING 2008, pp. 59]. MENDLING does so, in fact, using metrics to formalize the patterns.

²⁹ A comprehensive overview is available at <http://www.workflowpatterns.com> (accessed 10.05.2009).

Complexity

Complexity often involves the difficulty of handling a system, as it is hard to estimate the outcome of an action (therefore, popular lore goes “never change a running system”) [LEE 2003]. Apart from these notions, complexity is sometimes defined as a degree of disorder [SHANNON & WEAVER 1998].

Like the approach by [CARDOSO 2006b], complexity is thus characterized by:

- **Structure:** A complex system is a potentially highly structured system which indicates a structure with variations [GOLDENFELD & KADANOFF 1999] [OLIVER et al. 1997, p. 29].
- **Configuration:** Complex systems have a large number of possible arrangements of their parts [KAUFFMAN 1993] [HOLLAND 1996] [BAR-YAM 1997].
- **Interaction:** A complex system is one in which there are multiple interactions between many different parts [RIND 1999].
- **Inference:** System structure and behavior cannot be inferred from the structure and behavior of its parts [KAUFFMAN 1993] [HOLLAND 1996] [BAR-YAM 1997].
- **Response:** Parts can adjust in response to changes in adjacent parts [KAUFFMAN 1993] [HOLLAND 1996] [BAR-YAM 1997].
- **Understandability:** A complex system is one that by design or function, or both, is difficult to understand and verify [WENG et al. 1999] [IEEE 1991].

The completeness of a solution (or rather the lack of determining the completeness of a complex solution), as well as uncertainty about a complex system’s state and coupling are not part of the definition, although they, too, are commonly related to complexity [DANILOVIC & SANDKULL 2002].

Process Complexity

In particular, the contextuality of a system (i.e., its strong relation to the environment) and radical openness (i.e., its possible unforeseeable interaction with entities not originally suspected to be relevant) are seen as the core drivers of complexity [CHU et al. 2003]. This implies that a complex system has many entities which have many different relationships, as stated for systems³⁰.

Processes are systems in themselves; CARDOSO points out that a process exhibits multiple facets of process complexity [CARDOSO 2005a]. He bases his conclusions on the similarities of executing a process to running a software, as it has structure (software is a highly structured system, cf. above), arrangements (a program has many possible arrangements), interacts with different parts, inference (the behavior of parts may be different than the sum of the parts), response (program gives response to input) and understandability (a software program is difficult to understand and verify). He thus transfers and adapts the work from software

³⁰ Further definitions of complexity can be found in [SUH 1999]. [WEBER 2005] discusses different ways of looking at complexity in engineering design.

engineering to describe process complexity as the number of tasks in the process (task complexity), the degree of cross-linking in the arrangement of tasks, and the related decision points such as AND, OR, or XOR (control-flow complexity), the degree of dependency of the information objects and their mapping to tasks, resources etc. (data-flow complexity), and the degree of interdependency of resources and their attribution to the tasks in the process (resource complexity).

Cardoso thus deduces that the definition provided by the IEEE Glossary is appropriate to describe process complexity [CARDOSO 2005b]. It is adapted here to include not simply the sequence of tasks, but all necessary supportive domains and their relationships.

Process complexity is “the degree to which a process is difficult to analyze, understand or explain. It may be characterized by the number and intricacy of activity interfaces, transitions, conditional and parallel branches, the existence of loops, roles, activity categories, the types of data structures, and other process characteristics” [IEEE 1991].

Focusing on the network of entities and relationships of the system “engineering design process”, in the following, different models, methods, and tools that are commonly used to understand, represent, and analyze such systems are presented. First, approaches to manage complex network structures are reviewed; then, the concept of structure is extended to engineering design processes in order to collect metrics that embody structure in a process.

In summary, processes can be understood as a special class of systems that are constructed from entities and their relations among each other. Within this network-like structure, certain patterns can be identified that serve as structural characteristics. To productively manage such a complex system, a detailed classification of entities and relations into appropriate domains and relationship types provides a systematic basis to model a system and to compare the entities among each other. The actual entities and relations can be taken from any process model, as will be shown in section 2.2.3.

There is a consensus that the structure of a system, i.e., the purposeful constellation of entities and relationships, drives its behavior. The identification of the patterns that repetitively occur within such a structure can serve as a basis to draw inferences about the behavior of a system. It is, therefore, necessary to collect these patterns and relate them to their structural significance to better analyze a system and understand its behavior. The structural characteristics that are available so far will thus be explained in the following sections.

Lastly, complex systems tend to have many domains that are not independent of each other. The aggregation of different views onto a system provides a compact means of generating manageable models that do not reduce the structure of a system but condense it to its individual impact on a domain of reference. It is, therefore, necessary to further explore the creation of such aggregate views to better identify structural patterns within large processes.

2.1.2 Graph Theory

Graph Theory³¹ provides the mathematical foundations to study any kind of network, called a graph. A graph is an ordered pair $G = (N, E)$, where N is a set of nodes (also called vertices) and E is a set of edges (also called arcs); being a 2-element subset of N , a graph is thus a formal description for “boxes and arrows” when drawing a network on plain paper. It is commonly understood to date back to the works of Euler and the “Seven Bridges of Königsberg”, i.e., the mathematical solution to the question whether a walk through the city of Königsberg could be routed in a way that all bridges would be crossed only once.

Graph Theory describes networks in a generic way, attributing to them the following basic properties:

- They can be directed (“digraph”) or undirected, or both (“mixed graph”).
- They can have a weight associated to nodes or edges (“weighted graph”).
- They can have loops (“simple graph”) or not.
- An edge can connect a node to itself (“loop”).
- They can have multiple edges between two nodes (“multigraph”), one, or none, or one edge connecting one node to many others (“hyperedge”).
- They can have edges not associated with any node (“half-edges”, “loose edges”).

Graphs have basic characteristics used to compute more complex analyses:

- Two edges of a graph are called “adjacent” if they share a common node; equally, two nodes are called “adjacent” if they are connected by an edge.
- An edge is called an “incident” (in a directed graph), if it is directed towards a node; the opposite direction is called an “outgoing” edge.
- The number of edges that connect to a node is called a “degree” of the node.
- Nodes are discernable (“node-labeled graph”), edges as well (“edge-labeled”). In particular cases, nodes or edges can also be treated as not distinguishable.

Graphs also contain certain basic structures that can be used to describe them:

- Elements in a graph can be “disconnected”, i.e., a node has no edge to any other node.
- A graph is “complete” if every pair of nodes is connected by an edge, i.e., if the graph contains all possible edges. Such a graph, in which every node is connected to every other node, is also called a “clique”.
- If a graph is “strongly connected”, it does not necessarily have any cliques in it, but every node can be reached from every other node.

³¹ There are many textbooks available on graph theory that need not be included here; for a better understanding, refer to [DIESTEL 2006]. [GROSS & YELLEN 2005] also is used as a common, very detailed, textbook; a good introduction in German is available by [TITTMANN 2003].

- A graph is “bipartite” if the set of nodes can be grouped into two sets U and V in a way that each edge has one node in U and one in V . This implies that the nodes in U are disconnected; equally no node in V is connected to another node in V .
- A “path” is a set of adjacent edges listed in a specific order; the path can be attributed by its length. A “walk” more specifically addresses an alternating sequence of nodes and edges that form an open or closed (=“cycle”) walk. The shortest path between two nodes is also called a “geodesic”.
- A “cycle” is a path that starts and ends with the same node.
- If a graph has no cycles, it is called a “forest”. If it is a connected graph that has no cycles, it is called a “tree”.
- A “spanning” tree is the minimal graph necessary to connect all edges in a graph.
- A “planar” graph is a graph whose edges do not cross each other.
- A “subgraph” is a graph S contained within a graph G : G is the “supergraph” of S .
- “Graph labeling” is used to assign integer labels to nodes and edges; this can be used for the “coloring” of a graph, assigning a color to each node with no tuple of neighboring nodes being of the same color.
- The “genus” of a graph refers to the fact that the graph can be drawn as a planar graph on a surface of genus n (e.g., a sphere, a torus, etc.).

Graphs are commonly modeled as “boxes and edges”. There are many methods to draw a graph, serving different purposes. A common algorithm is a force-based layout that arranges the nodes in a way such that nodes which are closely connected arrange as neighbors, repelling less connected nodes [FRUCHTERMAN & REINGOLD 1991].

Mathematically, graphs can be modeled as an Adjacency Matrix, which is similar to a DSM (see below). Adjacency Lists are also a possibility, listing which node is connected to what other node. MAURER ET AL. used this to indicate the usefulness of graphs to represent DSMs, while drawing equally on the extensive means of systematic analysis available in Graph Theory to extend analysis tools suitable for matrix-based methodologies [MAURER et al. 2004]. Other models, e.g., the Distance Matrix, are not considered here.

Commonly, the models and methods of Graph Theory provide the basis for analyzing structures, as shown in the next section (see [Table 2-2](#) on page 50). Graph Theory also provides the basic means of describing large networks in Network Theory.

2.1.3 Matrix-based methodologies to manage structures

Research on matrix-based complexity management³² has come a long way. Originating from a focus on process management using the **Design Structure Matrix (DSM)** [STEWART 1981], a whole community has developed around this research. DSM is able to model and analyze dependencies of one single type within one single domain. For example, for a product, the domain “components” can be considered. Using the relationship type “change of component 1 causes change of component 2”, an assembly can be analyzed with regard to the overall change impacts in order to model possible change propagations [KELLER et al. 2005]. [BROWNING 2001a] classifies four types of DSMs to model different types of problems: Component-, team-, activity-, and parameter-based DSMs. [JARRATT 2004] also proposes eight basic types of a DSM related to product architectures. However, all of these classifications present special cases of DSM application and should not understood as the only possibilities of modeling, but as suggestions for commonly accepted and useful models.

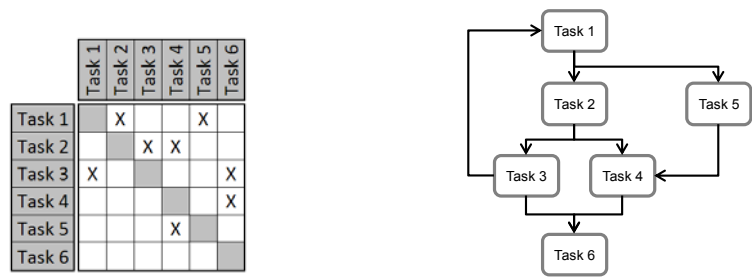


Figure 2-6: Binary Design Structure Matrix of a simple process

Binary DSMs only represent the existence of a relation, whereas numerical DSMs [BROWNING & EPPINGER 2002] implement a weighted graph to represent the strength of a relation. DSMs can either be directed (as shown in Figure 2-6), or non-directed; in the latter case, a DSM can either be written as a triangular matrix, i.e., only the upper or lower triangular matrix is used, or as a symmetrical DSM, i.e., the upper and lower triangular matrices are symmetric to the diagonal. Elements in DSMs are never reflexive, i.e., a relation from an element to itself is not permissible.

³² A recent compendium on different types of matrix-based methods is available as [LINDEMANN et al. 2009]. Similarly, [BONJOUR 2008] provides a complete overview of recent developments (in French). Also, the web portal www.DSMweb.org (accessed 10.7.2009) provides a reference data base related to matrix-based dependency modeling.

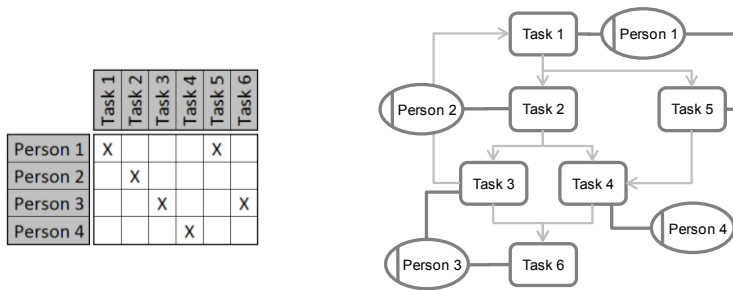


Figure 2-7: Binary Domain Mapping Matrix for the process shown in the previous figure

DSM was later extended to **Domain Mapping Matrices (DMM)** [DANILOVIC & BROWNING 2007]. The goal was to enable matrix methodology to include not just one domain at a time but to allow mapping between two domains [YASSINE, A. et al. 2003]. DMMs are thus rectangular, and again they can be binary or numerical.

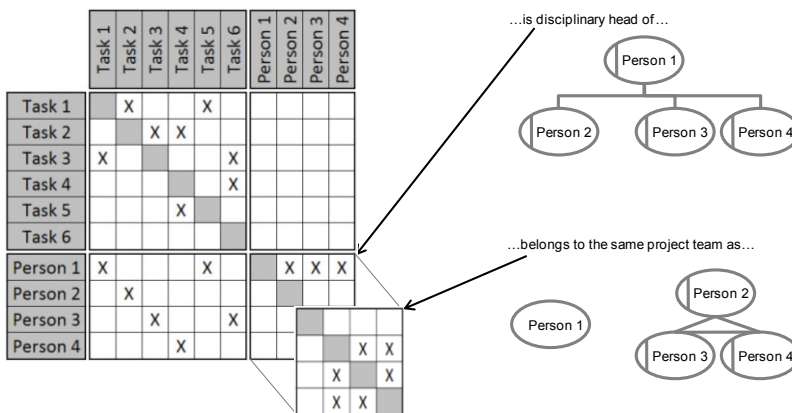


Figure 2-8: Multiple Domain Matrix combining the DSM and DMM from the two previous figures and introducing two additional DSMs for one domain, using two different relationship types

MAURER has taken this approach further to model whole systems consisting of multiple domains, each having multiple elements, connected by various relationship types [MAURER 2007]: the **Multiple Domain Matrix (MDM)**. Figure 2-8 illustrates this concept. It shows how a MDM basically is a DSM with more detailed DSMs along its diagonal and DMMs outside the diagonal. It also depicts how multiple relationship types create several representations of a specific submatrix of the overall MDM.

The MDM allows a system's structure to be analyzed across multiple domains, condensing each single analysis into one aggregated DSM that represents multiple domains at one time. That way, the MDM is able to apply algorithms for DSM analysis meaningfully across several domains [WALDMAN & SANGAL 2007] [CRAWLEY & COLSON 2007].

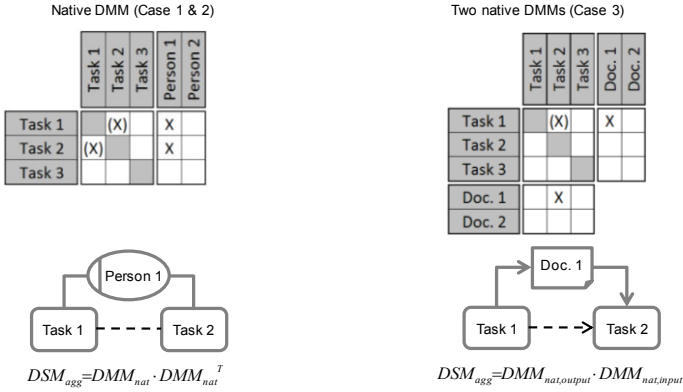


Figure 2-9: Possible cases 1 to 3 for computing an aggregated DSM within a MDM [MAURER 2007, pp. 113-116]

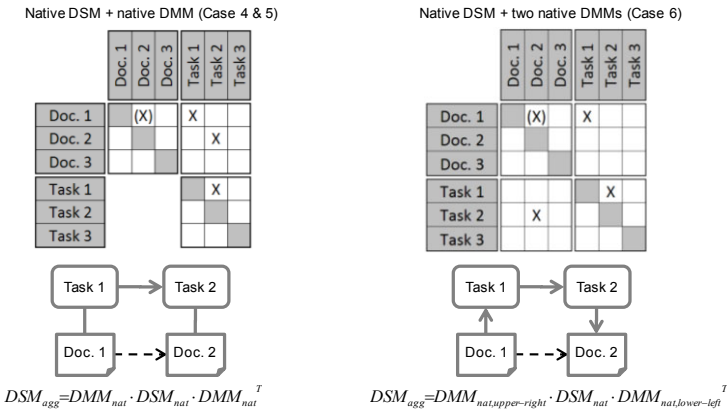


Figure 2-10: Possible cases 4 to 6 for computing an aggregated DSM within a MDM [MAURER 2007, pp. 113-116]

The main advantage of a MDM is that relations across more than one domain can be aggregated into a DSM using a domain mapping logic, using one native DMM (differentiating the direction of the DMM mapping as cases 1 and 2), two DMMs (case 3), one DSM and one DMM (differentiating the direction of the DMM mapping as cases 4 and 5), and ultimately two DMMs and one DSM (case 6). Figure 2-9 and Figure 2-10 illustrate these cases.

As it is very important “to determine which aspects are to be considered in order to answer a specific question” [MAURER 2007, p. 18], these cases can be used to collect and compile the existing information on the structure in accordance to the goal of the analysis. MAURER proposes two ways to do so: the opportunistic application (“see what you can get”) and the requirements-driven application (“define what you need”) [MAURER 2007, p. 93]. Each way is, at the same time, equivalent to the strategy of acquiring the data for the model. An MDM can thus be used either to gather native data or to combine data in a way suitable for an analysis. [BIEDERMANN & LINDEMANN 2008] extend the proposition and suggest a way of calculating cycles across domains of an MDM instead of elements of a DSM to calculate single DSMs out of a series of matrices in an MDM; however, this strategy remains largely unexplored.

There are several **strategies to analyze** the DSMs generated. Classically, a DSM is used for sequencing, tearing, banding and clustering. In sequencing, the rows and columns of a flow-oriented DSM are rearranged in a way that as few relations as possible remain below the diagonal, thus reducing the number of active feedbacks, leading to an ideal sequence. However, such an ideal sequence cannot always be found³³. Tearing consists of choosing the set of feedback marks that obstruct sequencing the DSM. The relations that need to be removed are called "tears". Banding rearranges the rows and columns in a way that blocks of parallel entities remain, which, for example, in a process can be executed independently of each other. Thus, a “band” represents a group of elements being active in parallel. Clustering is executed to find those clusters of entities that are mutually related. [Figure 2-11](#) provides an overview. MAURER explains all techniques in detail [MAURER 2007, pp. 225-240]. Similarly, there are algorithms for the systematic analysis of DMMs, mostly focused on clustering [DANILOVIC & BROWNING 2007] [McCORMICK et al. 1972].

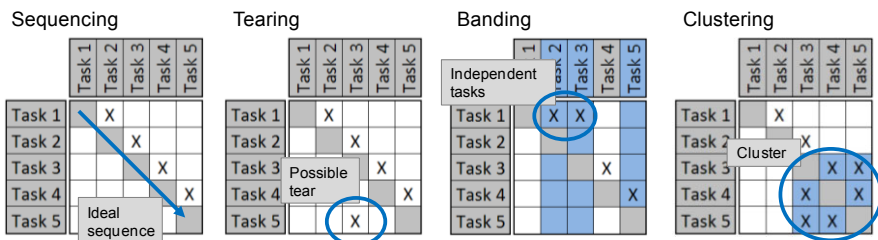


Figure 2-11: Classic DSM analysis techniques

Furthermore, DSMs can be analyzed for **structural characteristics**. The analysis is based on identifying patterns within a structure, i.e., typical constellations of entities and relations. A structural characteristic is defined as follows:

³³ See page 164 for the limits of triangularization.

A structural characteristic is a particular constellation of entities and relations, i.e., it is formed by a particular pattern formed from nodes and edges in the graph [CARDOSO, J. 2006a] [MAURER 2007, p. 123]. The characteristic gains its meaning by the way the pattern is related to the actual system it is part of, i.e., it must serve a special purpose in the context of the overall system [BOARDMAN & SAUSER 2006]. A structural characteristic only possesses significance in the context of the system it is describing.

Figure 2-12 categorizes the common basic structural characteristics that are currently available in different works. The structural characteristics shown form the basic building blocks to analyzing a structure. An overview is available in [MAURER 2007, pp. 225-240], therefore the classification is not detailed further³⁴. Eppinger similarly proposes that there are patterns across several domains [EPPINGER 2001]; however, he does not formalize them.

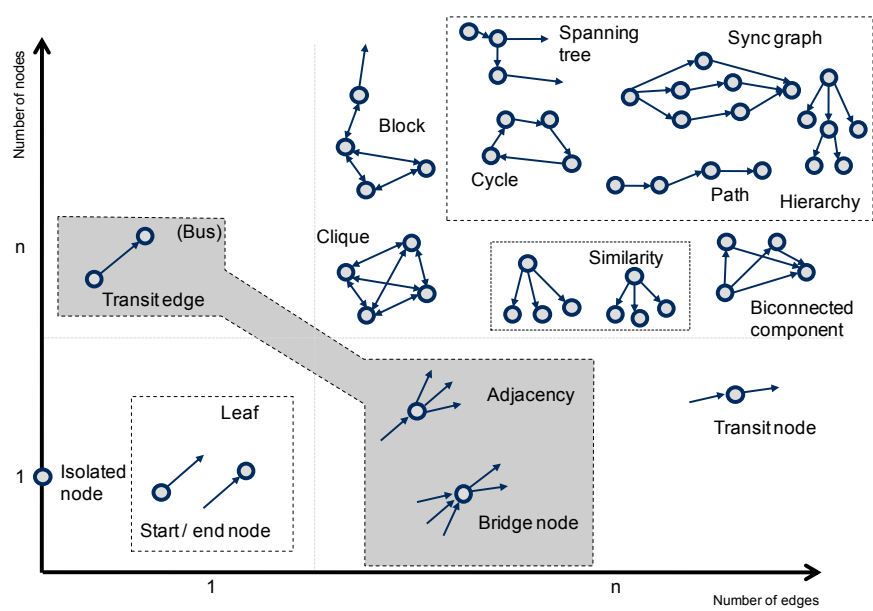


Figure 2-12: Overview of basic structural characteristics

Table 2-2 illustrates the available basic phenomena in graph theory [GROSS & YELLEN 2005] and shows which structural characteristics (Figure 2-12) they relate to. Although there is no complete one-on-one relationship between phenomena and structural criteria, the table regroups what phenomenon a structural criterion focuses on. For each, the table shows whether the mathematical phenomenon has an application in engineering design or not. As can be seen, there are a number of phenomena that have no application (yet).

³⁴ Structural characteristics will later be used to create structural metrics. They are detailed in section 5.1.

Table 2-2: Phenomena in graph theory and adoption as structural criteria (based on [GROSS & YELLEN 2005])

Graph Theory	Structural Criteria available	Structural Criteria still missing
Cliques, subgraph	Strongly connected components	
Walks	Cycles, paths, distance	
Trees	Leafs, roots, spanning trees, knots	
Adjacency and degree	Neighborhood, bridges, degree / activity, independence, connectivity /attainability	
Genus		Planarity, thickness
Weighted graphs and networks	Weighted nodes, weighted edges, minimum spanning tree, shortest path	
Coloring		Chromatic number, k-coloring, color-classes
Multipartite graphs	N-partite graphs	Disjunctive sets
Eigenvalues		Eigenspectra

There are still **shortcomings to matrix-based methodologies** [KREIMEYER et al. 2008a].

Generally, attributes to an edge are only possible to a limited extent. Whereas matrix-based models are mostly designed as qualitative models and not as quantitative models, this fact still hinders the transfer from other models such as Petri-nets into a DSM.

Linking elements of a matrix to a relation is not possible in a matrix. While two nodes of a system can be related, it is not possible to link an element of a matrix to a relation (i.e., a node to an edge). This issue is resolved in section 4.4.2.

So far there has not been any systematic research to generate a catalogue of structural characteristics. Equally, the differentiation between structural characteristics and structural metrics is still not clearly defined.

Some fields of research have developed indicators to measure the degree of complexity of a system as a numerical value. Yet little work has been done so far. While the metrics developed as part of this research are not exhaustive, they are meant to fill this gap (see section 5.2).

Network structures in practice often contain decision points that can only be represented with limits in a matrix. There has been work undertaken to overcome this problem [BELHE & KUSIAK 1995], which is not complete, however, as it does not allow logic operators to be included as part of the matrix description. Section 4.4.3 completes the existing approaches.

Common variant design is addressed through commonality issues, for example, [BRAUN & DEUBZER 2007]. The modeling of decision points and differentiation of co-existing solutions with a common base (e.g., one body with different component assemblies) still remains largely unsolved. In process management, process alternatives are often designed and compared to generate an improved

process. However, there is no methodology yet to generate such structures based on a common matrix-based description.

MAURER introduces a way of excluding several cells of a matrix upon certain conditions to describe boundary conditions [MAURER 2007]. Yet, matrix notation is still unable to work with more complex conditional settings; even for static, non-conditional relations within a system, no notation supports the use of existing algorithms.

The evolution over time (or another axis) remains unsolved, although many problems represented in matrices undergo changes, e.g., team structures; however, no real mechanism of evolution of a matrix has yet been found. A first approach has been shown [EBEN et al. 2008] based on Delta DSMs that map how one DSM differs from another [DE WECK 2007].

The management of hierarchical decomposition within a cell is still difficult to consistently describe [DANILOVIC & BÖRJESSON 2001]. Often, it is necessary to go into detail for a few cells only; while it is possible to zoom into such a matrix within one cell (it is very similar to an MDM), no description of how to handle the multitude of relations from the zoomed matrix going into one single row/column in the higher-ranking matrix is available.

The intuitive and graphical representation of MDMs is still unsolved. DIEHL proposes a 3D-visualization using several planes, where each plane represents a DSM, and the space in between is used to show how two planes interrelate [DIEHL 2009]. However, this is only applicable for small systems. Yet, there has been ample research for visualization³⁵.

Ultimately, no framework for a goal-oriented analysis has been proposed to operationalize the opportunistic and the requirements-driven approaches systematically [MAURER 2007, p. 93]. More generally, the potentials and limits of analyzing structural complexity in different contexts have not been given much attention. Section 6 proposes a framework to operationalize the selection of metrics for the context of engineering design processes.

In summary, different matrix-based dependency models were shown that enable the modeling and analysis of complex systems for which all entities and relations are basically known. Therefore, these models can be seen as deterministic models. In the context of analyzing a process chart, these models are, therefore, suitable to serve as a means to collect the entities and relations from any process chart in a common form, providing a basis to design structural metrics. In practice, however, these models are often not as definite as they seem, as generating a process model is essentially a process of consolidating different opinions as to how an as-is process is actually run. This, however, is not the focus here.

All modeling methods are paired with different analysis approaches that produce different structural characteristics which aid the analysis of the systems being modeled. Yet, a number of shortcomings still exist, some of which are highly relevant for process modeling, as section 2.2 will show. In particular, the modeling and analysis of logic operators in a structure, the systematic aggregation of

³⁵ Examples could be, for example, graph spectra [MAURER et al. 2004], linking it to common models [KARNIEL & REICH 2007] or various kinds of graphic representations [LIMA 2007].

different domains and their relationships, and the goal-oriented application of analysis in the context of process analysis have not yet been resolved.

2.1.4 Network Theory

Additional means of analyzing large network structures are provided by Network Theory. The combined use of approaches from Structural Complexity Management and Network Theory, therefore, provides a comprehensive toolset for the analysis of highly coupled structures; furthermore, both consider processes, among other fields of application, and thus provide empirical evidence for their use in process analysis. For a better understanding, the analysis tools provided by Network Theory and the relevant network models are presented here.

Network and Graph Theory are closely related. Whereas Graph Theory is focused on the formal modeling and analysis of the interaction of single nodes and edges of networks of limited size³⁶, Network Theory regards global properties of large networks. Thus, Network Theory makes extensive use of graphs, but with a different analytic approach³⁷ mostly based on statistics. Network Theory aims at creating viable models for large network structures, finding statistical properties to describe the networks, and making predictions about their behavior. The networks can be of any type. Commonly, four groups are regarded: Social networks (e.g., friendships, organizational structures in business, or email exchange), information networks (e.g., academic collaboration, websites and hyperlinks, or patent citations), technological networks (e.g., distribution networks, phone lines, or computer networks), and biological networks (e.g., metabolic pathways, genetic regulations, or the food chain) [NEWMAN 2003a, p.7]. To all these networks, the basic properties applicable to graphs (directed or not, weighted or not, etc.) shown in the previous section apply.

Network Science differentiates different models, as [Table 2-3](#) shows. The work on random graphs is mostly focused on models, as in [ERDOES & RÉNYI 1959]; generalized random graphs can be found, for example, in the works of [BOLLOBÁS 1981] and [NEWMAN et al. 2001]. Small world networks are commonly referenced in [WATTS & STROGATZ 1998].

Having these models available, the basic properties of large networks as currently available can be accounted for. In this context, emergence is an effect particularly important to Network Theory, acknowledging essentially the fact that certain patterns originate from even random networks, and it is these structural patterns that govern the behavior of the overall network.

³⁶ Graph Theory is not limited to a certain network size; however, a direct analysis of nodes is pointless for networks with millions of vertices, as commonly encountered in Network Theory [NEWMAN 2003a, p. 2].

³⁷ There are several review publications on the state of the art available: [NEWMAN 2003a] is the most complete, while [STROGATZ 2001] provides more examples. [ALBERT & BARABASI 2002] is also considered very comprehensive. [DOROGOVTSSEV & MENDES 2002], [HAYES 2000b], [HAYES 2000a], and [CAMI & DEO 2008] have reviewed the state of the art on Network Theory.

Table 2-3: Overview of statistical models for large, static networks (based on [NEWMAN 2003a])

Model	Basic idea of model	Applicability and critique
Random graph	Poisson distribution model: Uniform probability for a node to connect to another node	Mostly suitable for the theoretical observation of network properties.
Generalized random graph	Configuration model: Distribution of degrees is given, nodes connect at random	Ability to overcome the unrealistic degree distribution of a random graph, but inability to describe transitivity (also called "the strength of the weak ties", see page 54)
	Power law degree distribution model: Degree of nodes is distributed according to power law	
	Directed graph model: Degree distributions of degrees of incoming and outgoing edges	
	N-partite graph model: Similar to directed graphs, distributions are given for each domain of nodes	
	Degree correlation model: The correlation of the prevailing degrees in the network is given	
	Exponential / Markov Graphs: Few models available, only for special cases	
Small world model ³⁹	Random "shortcuts" across the network are created to improve the possibility of a node to reach another node by a short path	Suitable for most real networks; shortcuts often overrated in comparison to real world networks

The most common property is the **size** of a network, described by the number of nodes, the number of edges, the mean degree of a node, the mean distance of two nodes, and the diameter (i.e., the longest geodesic³⁸ in a network).

The **small world effect** formalizes the phenomenon commonly known as "six degrees of freedom", an experiment undertaken in the 1960s by Stanley Milgram to assess how many people a letter would pass through to arrive at a recipient unknown to the sender [KLEINFELD 2002]. Using the small world effect, the shortest path between two nodes of the network can be calculated as well as the mean path length of all paths connecting any two nodes in the network. Obviously, the more "small world" a network is, the quicker information is spread or a common opinion is generated (see [Figure 2-13](#)). Unlike matrix-based methods in engineering, however, the network models used have statistical properties that do not occur in technical systems. Thus, this effect is not directly transferable.

³⁸ A geodesic is the shortest path between a given pair of nodes.

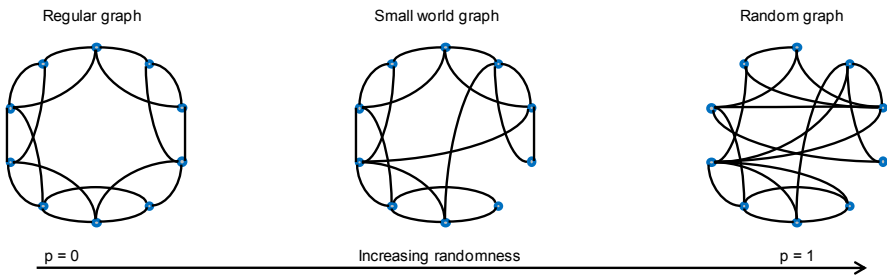


Figure 2-13: Different network types [WATTS & STROGATZ 1998]

The **transitivity**³⁹ of a network addresses the fact that if node A is connected to nodes B and C, it is quite probable that nodes B and C are connected, too. Thus, nodes with a high degree tend to generate clusters around them, even if these are not explicitly present [WATTS & STROGATZ 1998].

Resilience is also called connectivity. The research interest in a network is how the average path length across the network changes if individual nodes are removed, and when the network falls apart into distinct groups; additionally, the size of these groups is used to characterize a network [ALBERT et al. 2000]. Typically, large networks are resilient against the removal of random nodes. This relates to the fact that common network structures consist of nodes with a low degree, thus being rarely involved in communication [NEWMAN 2003a]. The targeted removal of nodes with the highest degree, on the other hand, quickly breaks down the connectivity of the networks to such an extent, that with a few nodes removed, the network falls apart. This is especially true for scale-free networks (see next paragraph).

Networks in the real world possess different **degree distributions** [ERDOES & RÉNYI 1959] [BOLLOBÁS 1981]. While a completely random graph with equal probabilities p of a node being connected will generate a homogeneous network, a scale-free graph will turn to a hub-and-spoke-like structure (commonly measured using a histogram of the degrees in the network). See Figure 2-14 for the two cases. The application is highly relevant to judge the robustness of a network in terms of the random failure of a node and its target (i.e., an attack). While networks tend to remain generally intact if a node that is minimally connected drops out, removing a hub may cause the network to fail completely, as, for example, the failure of the power grid in the USA in 1996 showed [WATTS & STROGATZ 1998]. Scale-free networks have received particular attention, following a power-law degree distribution. In fact, most large networks, e.g., the internet, are scale-free networks, having dedicated hubs and few connected spokes [LI et al. 2005] [KIM et al. 2002] [BARABÁSI & ALBERT 1999].

³⁹ Also referred to as “clustering”, which should not be confused with clusters in graphs or DSMs.

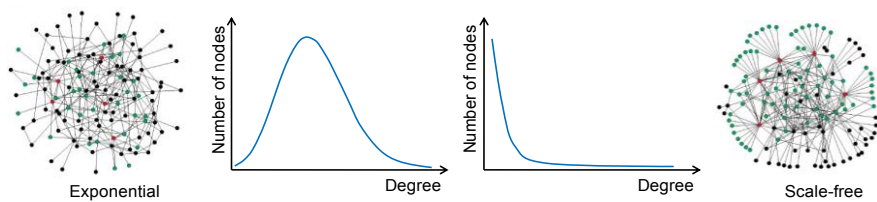


Figure 2-14: Homogeneous exponential and inhomogeneous scale-free network after [ALBERT et al. 2000]

In social networks especially, the characteristics of homophily or assortative mixing have led to research on **mixing patterns**. The interest is in how different nodes in a network establish groups of similarity of a kind that are not directly represented in the network [NEWMAN 2002] [NEWMAN 2003b]. A common example is a network of friendships in school. If the skin color of the children is introduced into the study, the network will sort the nodes in a way that distinct black and white subgraphs will appear [MOODY 2003]. A more special case is degree correlations; here, the pattern is about what constellations of degrees correlate [MASLOV & SNEPPEN 2002].

Social networks, in particular, show **community structures**, which are similar to the effect of clustering in DSMs (see Figure 2-11, right-hand side). A community is a group of nodes that are densely connected to each other and little connected to the outside network. To detect groups, cluster analysis is performed, which assigns a connection strength (much like the weight of an edge) to each pair of nodes [MOODY 2003] [NEWMAN 2003a]. However, more recent methods do not simply assign a community to a group of edges that are pairwise neighbors, but that are related via different geodesics across the group [GIRVAN & NEWMAN 2002].

Another feature of networks is the **navigation of the network**. While it is not only possible to find one's way across a large network even without knowing people explicitly ("a small world", [KLEINFELD 2002]), it is also common that individual paths across the network take shape.

Centrality is also a common property of individual nodes as well as networks [FREEMAN 1978]. Commonly, centrality is measured using the betweenness of a node, i.e., its position on a number of geodesics between all pairs of nodes in a network; the more geodesics a node is on, the more central it is because it is part of more communication paths across the network.

Finally, **motifs** have been recently researched. Motifs are parts of the network that, as a subgraph, are recurrent, i.e., they appear in a similar pattern in different places across the network [MILO et al. 2002].

In summary, Network Theory provides a set of structural characteristics that complement those provided by Structural Complexity Management. However, these structural characteristics are based on statistical network models that incorporate random network phenomena that do not necessarily occur in technical systems, such as shortcuts across the network, as found in Small World networks.

In fact, most networks that are regarded in Network Theory are networks from a social context. As processes are socio-technical systems, they do, in fact, exhibit these phenomena, too, to a certain extent. For example, an organizational structure in a process provides the basic communication structure; however, it does not elicit all communication channels available. The clustering coefficient that reviews the possible relations of a node is a good example, as it points to entities of the process that may be more importantly connected than shown in the actual network. Therefore, these structural characteristics provide a good complement to existing structural characteristics.

2.1.5 Other approaches to managing complex systems

General Systems Theory is interested in fundamentals, principles, and models valid for any kind of system, thus providing a general framework to describe how entities interact [VON BERTALANFFY 1968]. To this end, it applies differential equations. Systems Theory is based on the hypothesis of open systems, as opposed to many theories in physics, for example, i.e., systems that interact with their environment and are able to change their state through constant adaptation. General Systems Theory was later extended to New System Theory, differing mostly in the fact that the observer now was part of the system [PULM 2004, p.23]. It created the approaches of Self-Organization, Autopoiesis, and Dynamic Systems Theory. Self-Organization is, to some extent, related to emergence⁴⁰, i.e., how structure arises out of the relations and interactions of a system's entities. In particular, it regards how a system changes over time based on influences that originate from itself [DIETRICH 2001, p. 87]. Autopoiesis extends this concept to a level at which a system is able to reproduce itself based on the entities it is made of [MINGERS 1994]. Both approaches make use of four core principles inherent to open systems as proposed in General Systems Theory: Complexity (i.e., being a network of entities and their relationships), self-reference (i.e., the behavior of the system has an impact on the system itself), redundancy (i.e., entities that are in control cannot be separated from those that are being controlled), and autonomy (i.e., the behavior of the system can be detached – to some extent – from the environment) [PROBST 1987, p. 76].

Cybernetics [WIENER 1948] is closely related to General Systems Theory. Its main interest is in how complex, dynamic systems can be controlled to achieve a goal-oriented behavior [PROBST 1981, p. 7] regarding the control mechanisms among the entities of a system and the transfer functions that represent relationships. Cybernetics is an important foundation of management sciences [MALIK 2003, p. 80]. BEER bridged cybernetics and management science to create the field of Management Cybernetics [BEER 1972]. It is based on the concept of systems, and it incorporates the control mechanisms from cybernetics to model and determine how the actors in an enterprise are mutually interdependent and how they reach decisions by influencing each other [JACKSON 1991]. Cybernetics provides a paradigm where things are interdependent and certain organizational patterns directly impact the behavior of a system. This is also the basic understanding in this research.

⁴⁰ See section 2.1.4.

Systems Engineering is another discipline that uses the ideas from System Theory to better manage problems in engineering design. To do so, it extends the understanding of a system from the technical system under consideration to the project that creates the technical system [OLIVER et al. 1997, p. 85]. Systems Engineering makes extensive use of the network structure of the system and the control mechanisms that govern the system, and the context of Systems Engineering has given rise to a number of methodologies to model processes (e.g., IDEF, see appendix), Quality Function Deployment (see section 6.1.1), and various other models [INCOSE 2007]. A core concept is to link the behavior of a system (i.e., what a system does) to its structure (i.e., how a system is built), as proposed in this research [OLIVER et al. 1997, p. 21].

System dynamics equally regards systems, focusing on their internal network structure. However, it concentrates on the dynamics of this network to simulate its behavior in order to deduce improvement measures [FORRESTER 1977]. It is based both on quantitative models, mostly network-like flowcharts, and qualitative models to detect feedback loops that may either reinforce or balance the system.

Operations Research is interested in attributing resources to a problem under certain boundary conditions, using optimization algorithms to achieve an optimum solution. It applies methods such as linear programming, graph theory, scheduling algorithms, network theory, and different aspects of combinatorial analysis [FINKE 2008, p. ix]. Operations Research treats many problems that are similar in this research, and many algorithms for structural analysis have originated from Operations Research, especially regarding connectivity, shortest paths, matchings, and n-partite graphs [BIENA 2008].

Information Theory tries to quantify information, thus making possible many modern communication technologies. It applies a basic measure for the complexity of information, namely entropy, eliciting the average storage space necessary to recuperate a piece of information [SHANNON & WEAVER 1998, pp. 48-50]). It was later extended to Algorithmic Information Theory, combining it with the idea of determining the computability of an algorithm using a Turing Machine (i.e., a model of the computation of an algorithm). Tuning machines use the more formal Kolmogorov complexity as a measure for the computational resources necessary to describe a piece of information [BURGIN 1982].

To explain networks of economic transactions, **New Institutional Economics** is also related to the research presented here; however, it is mostly focused on explaining the rationale behind why two or more entities form a network. The Principal-agent problem regards, in particular, the “asymmetric” exchange of information within a network of potential partners, whereas Transaction Cost Theory focuses on the cost of exchanging information, and how this exchange can be organized most efficiently [KEIJZER 2007, p. 28].

2.1.6 Summary

With structure defined as the purposeful patterns that occur in the set of entities and relationships of a system, structural complexity prevails in different disciplines, creating different dependency models. Matrix-based models such as Multiple-Domain Matrices have only lately matured to a level able to describe

large systems involving several domains and relationship types, and many questions remain unsolved. Yet, the method is able to embed many analyses that can be traced back to Graph Theory or Systems Theory.

Table 2-4: Summary of features of structural analysis in different disciplines

	Structural feature	Meaning
Graph Theory	Adjacency	Immediate neighboring of two nodes
	Connectivity	Integrity of the overall network
	n-partite-ness	Existence of distinct, disconnected groups within the networks
	Paths	Channels of navigation through the network
	Cycles	Paths that end at their start node
	Reachability	Existence of at least one path to another node
	Planarity	Representation of network with no edges crossing each other
DSM	Sequencing	Ideal sequence of nodes in flow-oriented network
	Tearing	Iterations that inhibit an ideal sequencing
	Banding	Groups of independent nodes
	Clustering	Mutually related nodes
Structural Characteristics (MDM)	Isolated node	Node that is disconnected from the network
	Leaf	Node that is connected via only one edge
	Transit edge	Edge that lengthens a path without adding structure
	Transit node	Node that is transited by a path without adding structure
	Bridge node	Node that connects two structural characteristics
	Splits/joins	Node with few-to-many correlation of adjacent edges
	Bus	Combined split and join
	Hierarchy	Set of reachable nodes from a given node
	Similarity	Set of nodes similarly connected to rest of the network
	Biconnected component	Set of nodes that can be reached via at least two paths
	Spanning tree	Representation of minimum necessary reachability of a network
Network Theory	Size	Extent of network
	Small World Effect	Existence of shortcuts across network
	Transitivity	Probability of connectedness of neighboring nodes
	Degree distribution	Existence of hubs and spokes in network
	Mixing patterns	Relation of clustering to further attributes of the network
	Navigation	Relevance of shortcuts in Small World Network
	Centrality	Integration of a node into functioning of the overall network
	Motifs	Fractal patterns across different levels of abstraction

Graph Theory and Network Theory have evolved in parallel, creating different techniques to systematically analyze structural characteristics, such as the centrality of an actor in a social network or planarity as a measure of understandability of a system. The recombination of the different appropriate structural characteristics from the different disciplines provides a comprehensive toolset for the analysis of any system, as shown in [Table 2-4](#) (features appearing in several disciplines are listed only once). In particular, research in engineering and social sciences provides empirical evidence of the applicability of different structures in any kind of system and in processes specifically.

Furthermore, Multiple-Domain Matrices make it possible to create aggregate views that recombine different domains and relationship types into compact Design Structure Matrices used to analyze the long-range effects across several domains. The combination of Multiple-Domain Matrices with structural characteristics available in the different sciences, therefore, provides a comprehensive analysis tool for large complex systems. Yet, some gaps in modeling methodology still exist, especially the modeling of logic operators, the goal-oriented analysis and the aggregation of different domains. These issues will, therefore, be addressed later to complete the existing modeling methods.

2.2 Structural aspects of process management

Processes have been of interest for a long time. Process orientation was already proposed in 1934 [NORDSIECK 1934], yet process management did not really catch on until the 1980's [BECKER et al. 2005, p. 3]. A process-oriented organization is characterized by customer orientation, fewer interfaces, lower effort of coordination, clear responsibilities in terms of the results of the process, systematic improvement of process performance, process controlling, decentralized management, and organizational learning [SCHMELZER & SESSELMANN 2006, pp. 68-71].

In the following, the basics of process management are explained. In particular, those aspects that relate to the structure of a process are focused, i.e., dependencies and their patterns in process management. Quantitative analyses (e.g., towards lead time) are omitted.

2.2.1 Processes in Engineering Design

Process orientation has led to many different areas of research. In general, the more general notion of a process can be broken down into business processes and engineering design processes. Such a basic classification is, of course, not complete; there are many other taxonomies available, e.g., in primary (i.e., value-generating), secondary (i.e., doing the preliminary work for the value generation), and supporting processes (e.g., administration) [BECKER et al. 2005]. However, these are not addressed in this book, as the primary focus is on structure.

[Table 2-5](#) lists common definitions of the term “process”. Basically, a process is a set of interdependent tasks. Yet, a process is commonly characterized by its objective, its activities, its inputs and outputs, the events before and after it, its

reference to time, and the resources used. Thus, many different aspects collectively enable a process, for example, resources or the inputs and outputs.

Table 2-5: Definitions of the term "process" in literature

Reference	Definition
[BECKER et al. 2003, p. 4]	"A process is a completely closed, timely and logical sequence of activities which are required to work on a process-oriented business object. Such a process-oriented object can be, for example, an invoice, a purchase or a specimen."
[DAVENPORT 1993, p. 5]	"A process is simply a structured, measured set of activities designed to produce a specified output for a particular customer or market. It implies a strong emphasis on how work is done within an organization A process is thus a specific ordering of work activities across time and place, with a beginning, an end, and clearly identified inputs and outputs."
[HAMMER & CHAMPY 2003, p. 35]	"We define a business process as a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer."
[HARRINGTON 1991, p. 9]	"A process is any activity or group of activities that takes an input, adds value to it, and provides an output to an internal or external customer."
[VAN DER AALST & VAN HEE 2002, p. 4]	"A process consists of a number of tasks that need to be carried out and a set of conditions that determine the order of the tasks. [...] A task is a logical unit of work that is carried out as a single whole by one resource. A resource is the generic name for a person, machine or group of persons or machines that can perform a specific task."

The following definition for a **process** used for this research is based on [VAN DER AALST & VAN HEE 2002, p. 4], while additionally introducing the aspect of a *network* of tasks that are highly interdependent, which is of high relevance especially to processes in engineering [O'DONOVAN et al. 2005]. This definition is used, as it embodies the concept of a process as a multi-layered network (i.e., the first hypothesis of this research) and thus lays the foundation of assessing the structure of a process to deduce indications about its behavior.

A process consists of interdependent tasks that exchange information via artifacts. The process is enabled and supported by the purposeful allocation of resources and time-oriented constraints. All of these entities are interrelated, on the one hand, via the input-output relationships among tasks along the principal process flow, and, on the other hand, via other relationship types that generate the overall process network.

In this context, a task is a logical unit of work that is carried out as a single whole by one resource over a period of time. A resource is the generic name for a person, machine, or group of persons or machines that can perform a specific task. All entities may have relationships among themselves.

A **business process** is “a special process that is directed by the business objectives of a company and by the business environment. Essential features of a business process are interfaces to the business partners of the company (e.g., customers, suppliers). Examples of business processes are the order processing in a factory, the routing process of a retailer, or the credit assignment of a bank” [BECKER et al. 2003, p. 4]. Such a process is, therefore, repeatable without the necessity to generate knowledge about the process execution.

An **engineering design process**, in contrast, is a process during which knowledge about an object is generated. As this object still necessitates designing, its nature is – at least in part – unknown. This generates uncertainty throughout the process that needs to be managed, and that causes an engineering design process to be much less deterministic than a business process. Table 2-6 distinguishes business processes and engineering design processes.

Table 2-6: Difference between business processes and engineering design processes [VAJNA 2005, p. 371]

Business process	Engineering design process
- Processes are fixed, rigid, have to be reproducible and checkable to 100% - Results have to be predictable - Material, technologies, and tools are physical (e.g., in manufacturing) and / or completely described (e.g., in controlling) - Possibility of disruptions is low, because objects and their respective environments are described precisely - No need for dynamic reaction capability	- Processes are dynamic, creative, chaotic; many loops and go-tos - Results are not always predictable - Objects, concepts, ideas, designs, approaches, trials (and errors) are virtual and not always precise - Possibility of disruptions is high because of imperfect definitions and change requests - There is definitive need for dynamic reaction capabilities

As can be seen, in engineering, design processes have the character of problem solving [LINDEMANN 2007, pp. 45-47], i.e., they cannot simply be processed but necessitate the generation of knowledge [HATCHUEL & WEIL 2003]. They thus represent a “wicked problem”⁴¹ [RITTEL & WEBBER 1973]. Mostly, this is due to the high degree of novelty that is common for any product being designed. As a result, during the process, there is always a high degree of uncertainty present about the outcome – the earlier in the process, the more uncertainty there is in the process [LORENZ 2009, pp. 27-30]. In process management, this uncertainty takes shape especially in iterations, during which the design is reworked, improved and refined [WYNN et al. 2007] [ROELOFSEN et al. 2008]. Often, too, these iterations are not regularly cyclic, but they occur as leaps forward or backward in time [BADKE-SCHAUB & GEHRLICHER 2003]. While the process often creates quite

⁴¹ A “wicked problem” refers to a problem that cannot be definitively described and that has no definite solution. Therefore, there are no optimal solutions, and solving the problem is hardly possible, only indications can be given.

erratic patterns due to this, artifacts, or rather the knowledge about their desirable properties, are characterized by their growing concretization during process runtime [KREIMEYER et al. 2008c], and the intermediate results that represent certain stages of this concretization determine the path of the process [VAJNA 2005] [PONN & LINDEMANN 2005], at times necessitating the re-planning of the process. At the same time, engineering design processes are often impacted by moving targets and late changes to the initial concept due to late learning during the design process. Overall, engineering processes, therefore, have a low degree of repeatability [VAJNA 2005], and they are difficult to model and plan [O'DONOVAN 2004]. Yet, their behavior follows basic patterns [EPPINGER 2001], namely the specific mutual dependencies between the organization, the process and the product architecture. These need to be well aligned and mutually adapted.

Engineering Design Processes are often carried out as **projects**, with the project organization bridging the organizational hierarchy and the common process [LINDEMANN 2007, p. 12 and p. 16]. A project, however, is understood as a “temporary endeavor undertaken to create a unique product, service or result” [PMI 2003, p. 5]. WYNN differentiates processes as mechanistic or projects as non-mechanistic [WYNN 2007, p. 84]; the interest in the process is more on the mechanism (or structural patterns, as in this research), while a project plan is more focused on the timeline.

In this context, the **control flow** is an important aspect. The control flow (also called control view) is the set of relations of the various entities of a project [SCHEER 1999, p. 102]; it is thus equivalent to the network of entities in the process definition applied here. The concept generally prevails in business process management. However, the term is used by other fields of science, as well.

Process management makes use of this understanding to analyze, design, implement, enact, monitor, and evaluate processes to improve value creation in the enterprise [ZUR MUEHLEN 2004, pp. 82-87], shown in [Figure 2-15](#). The importance of the relations among the different entities of a process as a basis for the behavior of the process is commonly recognized; this has led to the understanding that improving the interfaces between different entities in a process provides the biggest leverage to obtain a more efficient process [RECHTIN 1991, p. 29] [WASSON 2006, p. 18] [FLURSHEIM 1977].

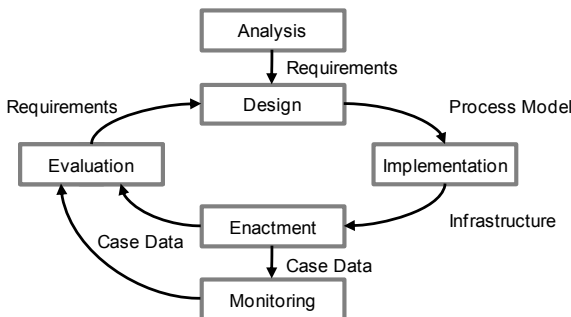


Figure 2-15: Business process management lifecycle [MENDLING 2008, p. 5]

Figure 2-16 visualizes the importance of how the paradigm, i.e., a certain perspective or understanding, drives the different activities during process management in engineering design [KREIMEYER 2008]; the model is similar to the Spiral Model in software development [BOEHM 1988].

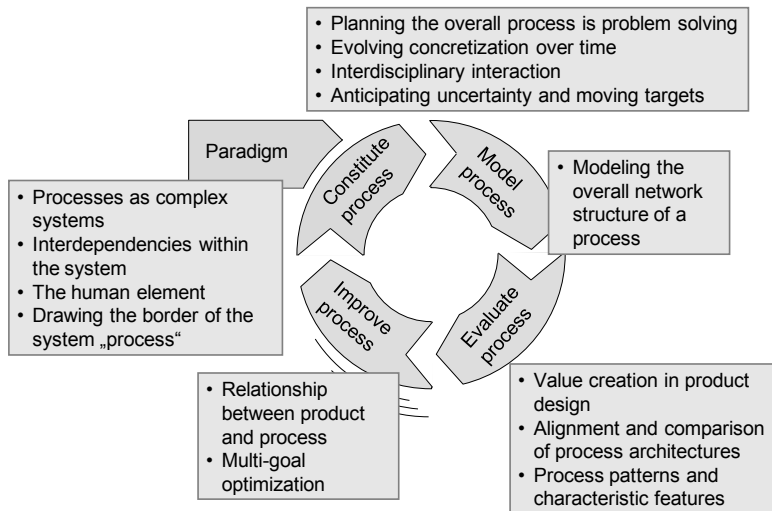


Figure 2-16: Influence of the modeling paradigm and continuous process improvement [KREIMEYER 2008]

The model works as follows: First, a model is prepared (the level of detail, the relevant views, the modeling language, and the access to information); then, problem areas of the process are defined and prioritized (i.e., the system border is drawn). Next, information is collected and models are created, and, last, the models are consolidated before they are analyzed for possible improvements [BECKER et al. 2005, 109-122].

In the context of this research, only structural aspects are of interest, and thus, the constitution only relates to collecting information about the relationships between the entities of the process. In the same manner, the model is a dependency model that then is evaluated, for example, for patterns that characterize the process's behavior, from which possible improvements are deduced.

As both figures show, the process model is the core component to process management [LINDEMANN 2007, p. 124], generating an overview about the current situation inside the company as the prerequisite for the improvement of a process. Of course, every process model is more or less simplified, made abstract, and reduced to the essentials. To model a process, usually the inputs and outputs, as well as the transforming tasks of the process, are captured, e.g., through workshops and interviews. The system boundary, in this context, describes departments, persons and facilities assigned to the system.

STETTER [2000] highlights his hypothesis that some weakness can be found in every design process model, for example, cost-intensive iterations in late phases, problems in finding and retrieving stored information, or problems caused by frequent product changes [STETTER 2000, p. 48]. Based on this hypothesis, it can be summarized that the identification of strengths and improvement potential in industrial practice not only consists of a search for strengths and weaknesses, but also includes the selection of the most prominent improvement potential.

The relevance of an improvement is given by the goals of a process improvement project, which dictate the modeling paradigm and which guide the later analysis of the model(s) created. The following section reviews possible goals more closely.

2.2.2 Goals of analyzing, improving and managing processes

Processes are managed for a number of reasons, satisfying different stakeholders, and there are various classifications of the concepts and goals of process management. Table 2-7 lists those aspects related to the structure of the process. These are adapted from the literature on typical errors, common problems, or the general intent of process management. Their categorization, as shown in the left-hand row, can be understood as common goals that for which processes are analyzed and improved.

The table is constructed from the references shown in the top row. From each reference, relevant concepts in process management were collected. In fact, some references directly address the goals of process management [KREIMEYER et al. 2008b] [BECKER et al. 2005, p.5, p. 30, p. 124], while others speak about the foci of process improvement in a more general manner [ZIMMERMANN 2008, p. 72] [SCHMELZER & SESSELMANN 2006, pp. 68-70] [IDS SCHEER 2007, p. 10] [GAITANIDES et al. 1994], and again others address typical problems in processes [BEST & WETH 2009, p. 77] [EUROPEAN FOUNDATION FOR QUALITY MANAGEMENT 1995]. All of these concepts were collectively classified with regard to their structural content, i.e., only those concepts that relate to the structure of a process to at least some extent were kept. In the context of this research, the concepts shown will be used as a framework to systematize a process in a goal-oriented manner. Section 6.1 will show common methods into which these concepts can be embedded.

Table2-7: Different concepts in process improvement

Concepts of process management	[ZIMMERMANN 2008, p. 72]	[BEST & WETH 2009, p. 77]	[KREIMEYER et al. 2008b]	[BECKER et al. 2005, p.5, p. 30, p. 124]
Planning	• Long runtimes	• Long lead times	• Speed of design process	• Critical paths • Time buffers
Resource consumption	• Effort for coordination • Redundant tasks	• Redundant work • Resource limits • Resource availability	• Optimum between time, quality, and resources	• Cost reduction • Obsolete processes • Administration
Quality	• Redundancy to intercept errors • Propagation of errors	• Fragmented tasks • Errors in paperwork	• Quality	
Flexibility				• Adaptation • Flexibility and robustness • Moveable activities
Organizational decomposition		• Hierarchical reporting • Coordination	• Design team coordination • Setup of teams	• Adaptation of capacities
Interfaces	• Inefficient interfaces • Insufficient supply of information	• Errors in transactions • Disruptions (media, resources) • Information: oversupply, outdated, incomplete	• Information exchange • Operations / workflow management	• Integration of participants • Optimization of interfaces
Transparency of process		• Complexity of content • High effort for maintenance	• Coordination of distributed design	• Process knowledge • Standardization of workflows
Planning	• Repetitive tasks • Iterations	• Expenses for coordination	• Optimization of internal processes • Automatization of processes	• Speed

Table 2-7: Different concepts in process improvement (continued)

Concepts of process management	[EUROPEAN FOUNDATION FOR QUALITY MANAGEMENT 1995]	[SCHMELZER & SESSELMANN 2006, pp. 68-70]	[IDS SCHEER 2007, p. 10]	[GAITANIDES et al. 1994]
Resource consumption		• High performance of process	• Efficiency • Reduction of costs	• Use of methods • Efficiency
Quality	• Consistency of information		• Effectiveness	• Coherence • Robustness (i.e., tolerance towards errors) • Precision
Flexibility				• Flexibility
Organizational decomposition	• Responsibility	• Distinct responsibility		
Interfaces	• Interfaces within process	• Few interfaces	• Introduction of standards • Harmonization • Roles for collaboration • Collaboration with external partners	
Transparency of process				

2.2.3 Process models and their structural content

Process models are an essential part of process management, as they help in understanding a process by representing the entities involved, their relationships, and the quality of their interactions. They are thus used for a variety of purposes which coincide with the different goals of process management, as shown before.

To assess process models for their structural content, it is necessary to understand to what extent each individual process model depicts a part of the structure of a process. In the interest of comparing what model is made for what purpose, BROWNING assesses different process models as to their focus and the different stakeholders interested in a process model [BROWNING 2009] [BROWNING 2008]. He concludes that while every model in his review is made for a different purpose, many models convey similar information.

To suit different needs, numerous methodologies for process modeling are available, and non-exhausted lists and comparisons are provided, for example, by

[BICHLMAIER 2000, pp. 43-61], [BAUMBERGER 2007, pp. 299-316], [SPATH & WEISBECKER 2008], [BROWNING 2009], [HEISIG et al. 2008], [LANGER et al. 2009], or [KARNIEL & REICH 2009]. [BROWNING & RAMASESH 2007] look more deeply into network-like process models, reviewing the literature published in the last decade. They conclude that, among other foci, the interaction of tasks and their impact on overall process improvement can be found in all existing models, yet needs more focus.

To be able to design a comprehensive approach that allows analysis of different process models in terms of their structural content, it is necessary to outline how process models can be compared and returned to one common denominator with respect to their structure. Both necessities are explained in the following section, starting with the latter aspect of reviewing how the interoperability of such models can be assessed to then analyzing models for their common structural content.

Comparing and recombining process models

To develop an analysis method that suits not one but all common process models, a structured basis for its design is needed. To do so, a modeling framework needs to be created that incorporates the particularities of common process models, i.e., their specific modeling constructs, such as their semantics and semiotics. This section reviews the necessary methodology by looking more closely at the results of research on the interoperability of process models.

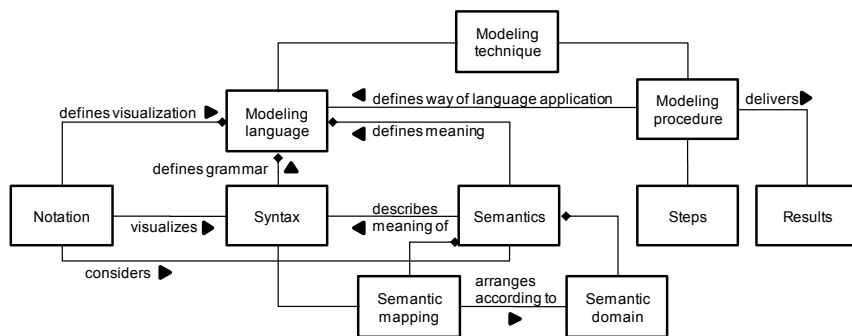


Figure 2-17: Aspects of process modeling [KÜHN 2004]

A process model involves many different aspects, summarized in Figure 2-17, for example, the semantics, their syntax, and the notation in which they are embedded. Existing process models vary in one or more aspects shown, and they do so to create an effective methodical support of one or several goals of process management. To do so, the semantics of the different modeling languages generally contain different aspects of, for example, scheduling, resource attribution or other domains. Other models, e.g., extended Event-driven Process Chains (EPC, see appendix 10.1.1), focus on the preparation and possible implementation of a process support through information technology and, therefore, integrate IT-systems and information objects into their semantics.

Again, YAWL (Yet Another Workflow Language, see appendix), for example, was designed to support the setup of the best possible workflow systems and, therefore, strongly focuses on formally correct decision logics, so-called workflow patterns, in the modeling scheme, while leaving out many other aspects that EPC integrates. However, if metrics are to be applied to estimate the complexity of a certain property of a process, it is necessary to have a basis to make these measures interoperable [CARDOSO 2005b].

As stated above, many process models are very similar to each other, following only marginally different foci, and a lot of work exists on the so-called **interoperability of process** or enterprise models.

To allow process models to be compared, BECKER & PFEIFFER [2008] suggest analyzing process models in terms of their semantics and syntax, and thus find two classes of conflicts that can arise between any two process models (language and ontology-based). To systematize the comparison of process models and to overcome these conflicts, HÖFFERER [2007] describes a meta-model-based approach to achieve a better interoperability between different process models by using ontologies⁴² to compare how “close” one process model’s meta-model is to the next [HÖFFERER 2007]. He thus proposes a terminological level at which models become comparable and which can be transferred from one model to another. The model is extended to a meta-level across the meta-models of different process modeling methodologies, referred to as a meta²-model (see Figure 2-18).

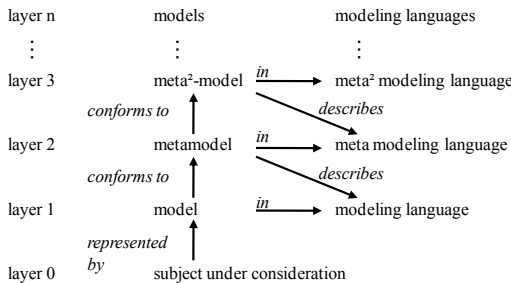


Figure 2-18: Different meta-levels to compare process models [HÖFFERER 2007]

Whereas modeling language-based conflicts limits the exchangeability of process models [BECKER & PFEIFFER 2008], the aspect of reviewing the semantics of processes at the level of the meta²-model is of interest to see if different process models offer similar structural content, namely similar domains and relationship-types. This approach is quite similar to the SEMAT methodology which compares

⁴² For the ontology-based comparison of process models, see also [GUIZZARDI et al. 2002], [PFEIFFER & GEHLERT 2005], and [SIMON & MENDLING 2007].

process modeling methodologies using a meta³-model as a framework for the comparison [KLUTH et al. 2008]. In the following, the meta²-level is chosen to compare the meta-models of different process modeling methodologies.

Comparison of common process models and their structures

With the goal of analyzing processes and their structure, this section reviews common process models as to their structural content. Common models are, in this context, those models that are either considered standard in industry according to [IDS SCHEER 2007] or that serve as a common reference in research with [BROWNING & RAMASESH 2007] as the main reference. The interest of this analysis is to generate a meta-model later for structural process modeling able to accommodate common process models (see chapter 4 of this book), thus becoming an analysis adapter for existing process models.

Table 2-8 represents the most common process models used for engineering design process analysis. All models are described in detail in appendix 10.1 of this book. In the appendix, each model is described as a flow-oriented model according to the individual modeling conventions, and it is represented as a specific meta-MDM that shows all domains contained in the respective modeling language as well as the relationship types that can be found. Figure 2-19 shows the SADT meta-model on the left-hand side with four basic types of activities that generate four domains in the meta-MDM on the right-hand side. Those domains that are related can be seen in the flow-chart model; their relationship types are given in the meta-MDM to the right-hand side of Figure 2-19.

Table 2-8: Common process modeling methodologies

Process modeling methodology	Acronym
Extended Event-driven Process Chains	eEPC
Object-oriented Event-driven Process Chains	oEPC
Business-Process Modeling (Integrierte Unternehmensmodellierung)	IUM
Unified Modeling Language	UML
Structured Analysis and Design Technique	SADT
Integrated Definition Method	IDEF0 / IDEF3
Business Process Modeling Notation	BPMN
Yet Another Workflow Language	YAWL
P3 Signposting	
Petri-Nets	
Process Module Methodology	PMM
Program Evaluation and Review Technique	PERT
Objektorientierte Methode für die Geschäftsprozessmodellierung und -analyse	OMEGA

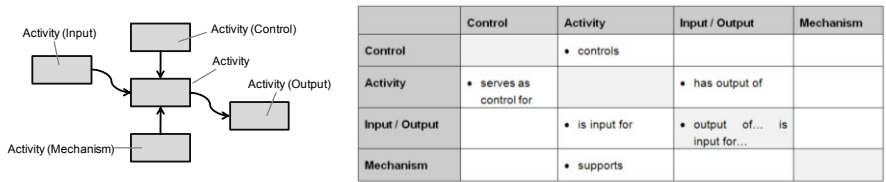


Figure 2-19: Example of a SADT as a flow-chart meta-model and as a meta-MDM to show the structural content

Using the meta²-level, the process models from Table 2-10 were compared concerning their structural content. Figure 2-20 shows the general approach, using an example of two partial process models in EPC and IUM on the lower level, their meta-models on the middle level, and the regrouped domains on the top of the figure. The relationship types are not shown in the figure but are, of course, part of the analysis, as well. The analysis consists of two steps. First, common domains are collected to constitute the domains of a meta²-model. Then, relationship types among these domains are established by collecting common relationship types of the individual meta-models for all process modeling methodologies.

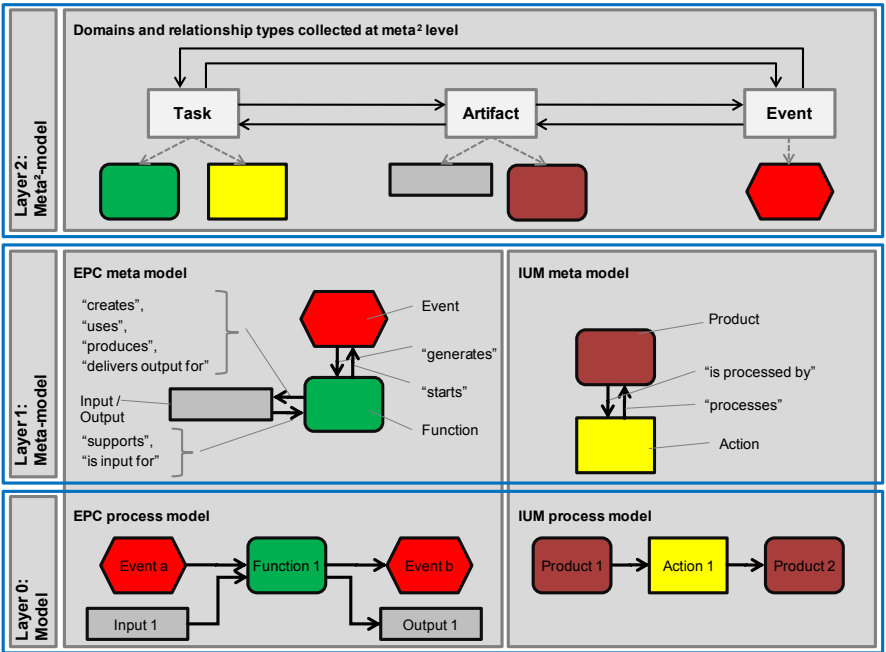


Figure 2-20: Example of the meta²-model approach taken here to compare process models and their structural content, aligning an EPC and an IUM Model at level 3 of the model by [HÖFFERER 2007]

Table 2-9 shows the domains of each process meta-model. Based on the descriptions of each model, these domains were regrouped into eight categories that represent common domains of process modeling. All descriptions and the necessary references are given in the appendix.

The table was designed like that of Table 2-7. Again, all individual classes of objects of the chosen process modeling methods were collected based on their descriptions in the references. For example, IUM lists “actions”, “functions”, and “activities” as descriptions for a task according to [MERTINS & JOCHEM 1998]. All classes of modeling objects were collectively classified. The resulting domains are listed in the top row of Table 2-9.

Table 2-9: Regrouped domains of structural interest for all 13 reviewed process modeling methodologies

	Task	Artifact	Control	Event	Org. unit	Res.	Time	Product
eEPC	Function	Input / output		Event	Org. unit	Resource	Milestone	(Output view)
oEPC	Method	Object		Message		Attribute (resource)	Attribute (object variable)	
IUM	Action	Product		Order		Resource		
UML	Activity			Initial / final node	Responsibility			
SADT	Activity	Input / Output	Control			Mechanism		
IDEF3	Unit of behavior	Objects / object states		Label				
BPMN	Activity / gateway	Data object		Event	Pool / Lane	Pool / Lane		
YAWL	Task	Data		Start / end node				
P3	Task	Parameter				Resource	Time / duration	
Petri	Transition			Place				
PMM	Work package	Input / output information	Informat. object / document		Tool / method			
PERT	Task			Event, time				
OMEGA	Activity	Business object			Org. unit	Technical resource	Milestone	External object

There are specific modeling constructs that can be understood as additional domains, as Table 2-10 shows: P3 and PMM allow the specific decomposition of the tasks in the process, forming a domain of their own. Similarly, BPMN and YAWL introduce an individual domain to represent logic operations. BPMN furthermore models text annotations explicitly as a separate domain, although such annotations are common in all process models and, in fact, do normally not represent a specific domain of their own.

Table 2-10: Further domains that occur among reviewed process modeling methodologies

	Decomposition	Logics	Annotation
BPMN		(Gateway)	Text annotation
YAWL		Condition	
P3	Process		
PMM	Process chain		

The same collection and classification can be done for the classes of relations that each modeling method provides. However, as the descriptions in appendix 10.1 show, many process modeling methods are not very specific about the different relationship types. For example, for SADT (Figure 2-20) no description is given at all; therefore the actual meta-model and the references only indicate an input/output relationship type [MARCA & MCGOWEN 1988]. The result of this collection of relationship types is not shown here but is taken up in section 4.3 to construct a meta-model for structural process modeling that entails all relevant domains and relationship types.

In summary, all process models contain aspects of the structure of a system to some extent, as they all consist of “boxes and arrows”. While some models specify the structure very strictly, others leave more room for process modeling experts to adapt the model. Yet, it is possible to review existing models at a more abstract level to find a common denominator, both with regard to the domains involved and their relationships. Section 4.2 takes up that concept to generate a meta-model that later serves as a basis for the application and interpretation of structural metrics for engineering design process analysis.

2.2.4 Strategies to analyze design processes and models

Process analysis is a common buzzword and a wide field of research⁴³, and it has been so for many decades now. Generally, there are specific methods to analyze

⁴³ For example, [CHAMPY & HAMMER 2007], [DE BRUIN et al. 2000], [BECKER et al. 2005], [SCHMELZER & SESSELMANN 2006], and [GAITANIDES et al. 1994] provide an overview of general process management. [CLARKSON & ECKERT 2005] and [FAHRWINKEL 1995] review approaches specifically for engineering design.

and deduce improvements for each individual aspect of process management, as shown in Table 2-7. There are also two kinds of strategies: continuous improvement and radical reengineering [HAMMER & CHAMPY 2003]. Both, however, are based on modeling a process and the analysis of the process models.

Engineering design is closely connected to problem solving [LINDEMANN 2003] [LINDEMANN 2007]. In turn, a process must support the best possible problem solving, and, therefore, the structure of the process organization needs to be closely connected to the product architecture.

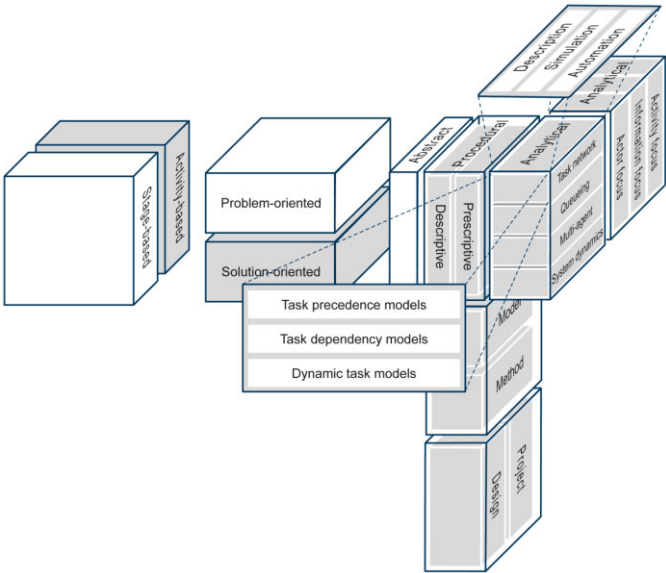


Figure 2-21: Framework of organizing process models and their analysis [WYNN 2007, p. 60]

This ambiguous borderline between process improvement and product design makes it difficult to systematize approaches to process improvement in engineering design. WYNN organizes process models, including the modeling and improvement intents in a framework consisting of several layers [WYNN 2007, pp. 16-60], as shown in Figure 2-21. The classes represent specific views of the design process, and each has created different models and improvement approaches.

As Figure 2-21 shows, the structure of a process is only addressed implicitly, which reflects the fact that there has been minimal attention in research on process improvement so far. In Wynn's system, structure is a part of the analytical approaches, and is addressed to activities, information transfer, or the actors. To this end, different strategies exist (task networks, queuing models, multi-agent simulations, system dynamics), but there is no overall approach that supports the selection of any of these quantitative, detailed and labor-intensive methodologies from a more qualitative point of view.

Generally, different methods of analyzing and improving processes are available [FREISLEBEN 2001, pp.71-74]: Experiential, analysis of historic structures, identification of duplication of work, analysis of interfaces, capability of communication, calibration of milestones, and comparison to reference models. However, these do not relate to structures directly except for the analysis of historic process models, which are the focus of this research.

The structure of processes has generally been addressed from a semantic point of view, having created many procedural models⁴⁴ to guide the design process; however, these remain at a very rudimentary level. At a more sophisticated level of detail, a DSM is generally used to interrelate the tasks of a design process to improve sequence, communication, or synchronization [BROWNING 2002] [KUSIAK et al. 1995] [STEWART 1981].

Other research has enriched the simple dependencies in these models to include, for example, the possibility to overlap tasks [KRISHNAN & EPPINGER 1997], the uncertainty inherent in a task [CHALUPNIK et al. 2008], decision options and scenarios [CLARKSON & HAMILTON 2000] [WYNN 2007], the different aspects of behavior of the actors of the tasks on a strategic level, such as coordination techniques, cooperation techniques, organization models, project management, and others [WHITFIELD et al. 2000], the behavior of actors on an operational level, e.g., resource attribution, scheduling, and others [COATES et al. 2000], and complex simulation [KARNIEL & REICH 2009].

Besides these approaches, which are focused mostly on the principal process flow as a set of tasks and their supportive entities (e.g., resources), another stream of research has focused on how the design process can be best aligned for the product architecture [SOSA et al. 2004b] [KREIMEYER et al. 2007c] [DANILOVIC & BROWNING 2007].

Little work has been done outside these basic metric designs, which relate mostly to product complexity. However, individual work on, for example, customer integration [KAIN et al. 2008], the role of social networks [LIBERATI et al. 2007], or enterprise architectures [WALDMAN & SANGAL 2007] have shown the need to manage the structure not only among the tasks but across all entities and domains of a design process [EPPINGER 2001].

So far, no comprehensive approach is available for the structural analysis of a complete process, and there seems to be no framework for the systematic improvement of a process's structure. There are, however, many fragmented methods and algorithms available that support the analysis and improvement.

2.2.5 Summary

A process can be understood as a system made up from different domains and relationship types, and it forms a network structure which only serves its purpose as a whole. While common business processes have already become highly

⁴⁴ Typical procedural models are, for example, the VDI 2221, the Munich Procedural Model [LINDEMANN 2003], the SPALTEN model [ALBERS et al. 2005], and the V-model of the VDI 2206. An overview can be found in [BAUMBERGER 2007, pp. 72-77].

complex, in engineering design this complexity is even greater through growing specialization and the distributed generation of knowledge.

Process management offers different methodologies to improve such processes, and improvements follow certain goals. Those goals relevant to structural process improvement were reviewed and consolidated to serve as a basis to set up a framework to organize structural metrics for process analysis.

An important foundation of process management is the modeling of processes. Each modeling method brings with it different domains and relationship types for the set of common aspects regarded by process management. Using a meta²-model, it was shown that process models in research and industry have common structural content that can be used to access the process structure embedded in these models; furthermore, different process models can be combined and/or compared that way with regard to the structure of the process. Nevertheless, there is no comprehensive approach to analyze the structure of a process in a goal-oriented manner. Although different methods for process analysis exist, these remain largely unconnected.

2.3 Metrics to analyze the structure of a process

This section reviews metrics as a means of the systematic analysis of large systems⁴⁵. First, the foundations of measuring are introduced; then the different aspects of good measurement of the complexity of a structure in network, software, processes, and engineering design, are reviewed. In particular, existing structural metrics are reviewed in detail to increase the understanding of the existing basis for developing metrics able to characterize different structures in an engineering design process.

2.3.1 Basics and measurement foundation

Metrics⁴⁶ are a means of representing a quantitative or qualitative *measurable* aspect of an issue in a condensed form [HORVÁTH 2003]. This “measurement is a mapping of properties of empirical objects to formal objects by a homomorphism” [ZUSE 1998, p. 92]. As such, a metric is therefore intended to depict an actual situation in a reduced and efficient manner.

Measurement theory⁴⁷ [ZUSE 1998] provides the basis for the design of such metrics. It describes how a phenomenon can be measured by establishing mapping

⁴⁵ “Large” will not be specifically defined, as it addresses essentially the fact that a system has many entities and relationships that are complex and thus difficult to handle.

⁴⁶ Measurement theory also refers to scales and measures, which, in this context, will be used synonymously. In economics, the term “performance indicator” is used, as well; however, it is not applied here, as it implies rating a good or bad performance rather than a basic metric [KAPLAN & NORTON 1992].

⁴⁷ Measurement Theory is succinctly reviewed in [SUPPES & ZINNES 1963] (available at <http://suppes-corpus.stanford.edu/article.html?id=43>, accessed 9.8.2009) and in [LUCE et al. 1988]

of an empirical concept to a mathematical concept. Measurement foundation addresses three common problems [STEVENS 1946]:

- The *representation* problem addresses the fact that a numerical scale should represent the relations that prevail in the real world.
- The *uniqueness* problem assesses the invariance of a metric to basic mathematical transformations.
- The *meaningfulness* problem allows the possibility of drawing conclusions from measured value.

Besides these foundations, the *validity* of a metric is, of course, of particular interest. The goal is to see if a metric actually represents what is supposed to be measured. The validity of a metric, therefore, largely relates to the meaningfulness problem. It can be broken down into three aspects [MENDLING 2008, p. 106], visualized in Figure 2-22:

- *Content*: Is the full range of possible meanings of the object represented?
- *Criterion*: Is the measured aspect the correct one to represent the topic of interest?
- *Construct*: Is the criterion in line with theoretical reasoning?

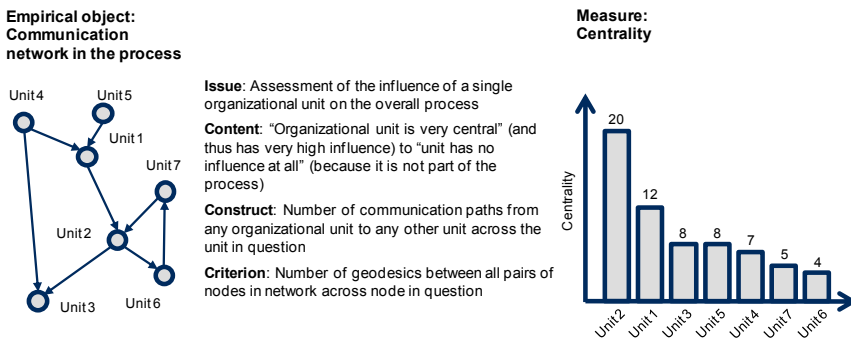


Figure 2-22: Example of a structural metric and the different aspects of its description

These problems are approached by different kinds of metrics. The scale hierarchy [STEVENS 1946] classifies metrics as nominal, ordinal, interval, and ratio scales. Identification numbers, for example, are *nominal* values, attributing a name to an empirical issue. *Ordinal* scales, such as the weight of an edge that relates to, for example, the intensity of the use of the communication channel represented by the edge, relate directly to the proportions in the empirical domain. *Interval* scales preserve only the relative distance between two empirical observations. *Ratio* scales, ultimately, extend the interval by a zero that expresses the absence of an empirical observation.

Metrics can also be classified as fundamental and derived [ZUSE 1998, p. 95]. Fundamental metrics commonly emerge at an initial point of research; derived measures aggregate one or several fundamental measures.

There is ongoing discussion about what kind of metrics can be seen as “real metrics” [AICHELE 1997, p. 74]. Some researchers argue that only ratios are real measures, as they not only yield a result but also a scale provided by the baseline of a fraction. Other researchers see the problem pragmatically, arguing that any metric that is able to express an empirical problem in a meaningful way is a useful measure. In this research, the latter view is followed.

Overall, two basic strategies to generate specific measures are possible: Either, well-understood fundamental metrics can be applied to a comparable context to generate new derived measures, or new fundamental metrics can be used. In this research, as presented in the following chapters, mostly new derived metrics are developed that use pre-existing empirical foundations.

A set of related metrics is commonly organized as a **measurement system**⁴⁸. The goal of establishing such a system is to organize metrics concerning their number, precision, and appropriate allocation based on their commonalities and relations among each other [AICHELE 1997, p. 79]. A measurement system can thus be defined as an “ordered set of metrics that are semantically related, that complement each other and that – as a set – are intended to represent an empirical issue in a well-balanced and complete way” [LACHNIT 1976].

A measurement system, therefore, is intended to structure a complex issue⁴⁹ in the real world in a condensed way, while making it possible to detail individual aspects as needed. Thus, an issue can be accessed in a structural manner, while it also serves as a framework to compare different objects under observation [SCHÜRRLE 1995, p. 14]. Like metrics per se, such a system intends to be homomorphous containing aspects that are of relevance in the real world. A measurement system can be structured in four different basic ways [AICHELE 1997, p. 81]:

- A *mathematical system* relates metrics by calculating combined or derived metrics from fundamental ones. This way, hierarchies of metrics, like the DuPont-System of Financial Control, are set up.
- *Practical systems* apply factual logic to relate metrics; relations are usually empirically established, such as the RL-System [AICHELE 1997, p. 81].
- A *heuristic system* is more focused than a practical system, developed explicitly to solve a specific issue and relate metrics to this issue. The GQM-approach shown in section 6.1.2 is an example that will be used later.

⁴⁸Also referred to as “scorecard”, “metrics suite”, or “ratio system” [REICHMANN & LACHNIT 1977]

⁴⁹ Measurement systems are most commonly found in economics, e.g., the balance sheet analysis, the DuPont-System of Financial Control, or Tucker’s Managerial Control Concept. Compare, for example, [AICHELE 1997, pp. 84-109] for an overview of measurement systems in economics or [GEIGER 2000, pp. 129-133] for such systems in engineering management.

- An *empirical system* is focused on statistically significant metrics that have originated from empirical observation. In contrast to a heuristic system, the attribution of a metric to an issue is considered more objective.

In practice, measurement systems are used as early-warning instruments, as means of analysis (e.g., for benchmarking processes or spotting improvement potential in a process), or as management tools to plan and control a complex system [GEIGER 2000, p. 99].

With the above definitions, TSAI and KERNLER summarize the **requirements of metrics** as follows [KERNLER 1996, pp. 35-38] [TSAI et al. 1986]:

- *Purposefulness*: Metrics should provide sufficient informational content to describe the issue in question.
- *Homomorphism*: Metrics should be designed to be as homomorphous with the behavior of original data as is possible and purposeful.
- *Simplicity*: A user who should be able to understand the metric easily.
- *Consistency*: A user should produce the same result when measuring the same process.
- *Automation*: The metric should be suitable for process automation.
- Metrics must be *additive*: If two independent processes are put into a queue, the value of the complexity metric should be at least the sum of the single values.

In addition, Weyuker's properties have become the established reference for metrics design, especially in software engineering [WEYUKER 1988]. They provide a set of properties that any good metric should fulfill. The set is, however, rather generic and is criticized for this [MENDLING 2008, p. 117]. In fact, a metric that is correct in terms of Weyuker's properties can still be meaningless, i.e., Weyuker's properties do not acknowledge the basics of measurement foundation [CHERNAVSKY & SMITH 1991]. Secondly Weyuker's properties deny the fact that a single metric cannot capture complexity in all its facets [ZUSE 1998]. Still, the properties are commonly applied to define metrics [CARDOSO 2005a].

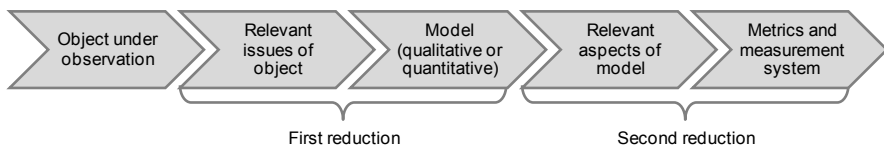


Figure 2-23: Modeling process of representing a system as a measurement system [GEIGER 2000, p. 95]

The **procedure to design metrics** commonly involves two stages of reducing a system [GEIGER 2000, p. 95]. In the first stage, the system from the real world is reduced to its relevant issues and then quantitatively or qualitatively modeled. In

the second step, the relevant aspects of this model are reduced and quantified. A similar model is proposed in [MUTSCHELLER 1996, pp. 63-83]; the author, however, extends the procedure to include the implementation and review of the applicability of the measures. In this research, essentially, the second reduction is the focus, as the metrics are used to analyze existing process models.

2.3.2 Metrics to describe networks

Metrics to describe network structures are generally derived from applied graph theory and network theory, as described in sections 2.1.2 to 2.1.5. They are usually grouped into three categories [BRANDES & ERLEBACH 2005]:

- *Element-level metrics* assess the position of a single element within the overall network.
- *Group-level metrics* regard the constitution and quality of groups⁵⁰ of elements within the overall network.
- *Network-level metrics* characterize the properties of the overall network.

MENDLING lists the most common metrics that are commonly used to describe a network structure [MENDLING 2008, pp. 107-109]:

- The *degree*, i.e., the number of edges connected to a node represents the connectivity of a single node.
- The *density* of a network, i.e., the ratio of existing edges to the maximum number of possible edges measures the cohesion of the overall graph.
- The *centrality* of a node, based on different methods [FREEMAN 1978], represents the cohesion of a network around a central node.
- The *connectivity*, i.e., the number of nodes that need to be removed for the graph to be unconnected, measures the level of homogeneity of the network.

Network metrics thus incorporate different aspects from Graph Theory; in fact, many of these metrics only take shape in different applied sciences, which will be reviewed below.

2.3.3 Metrics in software engineering

Software metrics are highly relevant to process management, as a software program and the control-flow graph of a process are very similar; several authors have drawn attention to the fact that executing a software is much like running a workflow or a process [CARDOSO 2005b] [GRUHN & LAUE 2006a] [ROLÓN et al. 2006a].

In software engineering, metrics are a popular basis for quality assurance⁵¹; they are employed to measure the degree of complexity in software to estimate the

⁵⁰ A module is a group of highly interconnected entities, whereas these high interconnections have been planned by the structural organization and are therefore predefined.

⁵¹ An overview of the foundations is provided in [NAVLAKHA 1987]; the recent state of the art can be found in [ZUSE 1998] and [DUMKE & LEHNER 2000]. An annotated bibliography is

level of error that the software is likely to encounter and to design test methods adapted to a specific new software development [ZUSE 1998]. Typically, thus, the metrics are selected top down [MENDLING 2008, pp. 110-111], i.e., the metrics are applied to measure certain aspects of a software program that are part of the quality assurance. This is why they are generally regrouped using the Goal-Question-Metric measurement system that will be shown in section 6.1.2.

Table 2-11: Common structural metrics for procedural programming paradigms

Metric	Description
<i>Lines of code</i> [AZUMA & MOLE 1994]	Lines of code measure the size of a program. This size varies substantially among different programming languages; thus the measure is of limited informational value (compare a dictum by Bill Gates: “ <i>Measuring programming progress by lines of code is like measuring aircraft building progress by weight.</i> ”).
<i>Cyclomatic number</i> [McCABE 1976]	The Cyclomatic number provides the number of paths through a program to visit all nodes, thus assessing the number of decision points. It is mostly used to define test cases. The original publication mentions a threshold of 10, beyond which the complexity should be justified. Microsoft's Visual Studio 2008 reports a violation for values exceeding 25.
Halstead's metrics: <i>length, vocabulary, volume, difficulty, effort</i> [HALSTEAD 1977]	Halstead disconnects the complexity of a program from the complexity of the language (number of distinct operators) the program is written in (total number of operators used); he also differentiates – in the same way – the variables (operands). Yet, only lexical complexity is measured.
<i>Information flow metric</i> [HENRY et al. 1981]	The information flow metric measures the fan-in and fan-out of a module, i.e., the degree of a connection of a module to the outside world. The metric is commonly used to estimate the likelihood errors.
Oviedo's <i>data flow complexity</i> [OVIEDO 1980]	The data flow complexity displays the degree to which a program can be broken down into distinct blocks that are executed as a unit and only have relations at a definition-level. It shows, thus, the potential for decomposition.
<i>COCOMO metric</i> [BOEHM 1981]	COCOMO is a cost estimation approach that uses the programming effort estimation metric as a basis for cost estimation. This metric is based on the size of a program (i.e., lines of code) and the programming language.
<i>Defect density</i> [ZUSE 1998, pp. 36-40]	The defect density represents the number of defects found in each module of a program. It is most helpful to locate error-prone parts of the software and to concentrate programming effort (as outliers among all modules).

Here, only basic metrics are presented, which will be of interest later. Commonly, there are two kinds of metrics, as there are two fundamentally different programming paradigms: Procedural programming uses a series of function calls, plus connecting split and join operations (e.g., goto, for...then, etc.), to constitute the control-flow of a program, whereas object-oriented programming uses classes

to define and instantiate objects, which then use methods to transform data. Thus, their control-flow graphs vary substantially.

Table 2-11 lists common metrics for procedural paradigms from the literature. They are typically based on counting function calls along the control flow; this is in line with the programming paradigm, as the metrics are generally oriented to mimic the execution of the program.

For object-oriented programming, the cohesion (i.e., the degree of the functional relationship of one entity to another entity) and the coupling (i.e., the interdependence between two entities) are important [WAND et al. 1990] and drive the metrics as shown in Table 2-12.

There are many more metrics available in software engineering. However, most of them are more conceptual and have not made their way into design practice. Two other measures, however, more abstract metrics have influenced many other metrics substantially. The *entropy* of information is part of information theory [SHANNON & WEAVER 1998] and measures the degree of disorder in a system; it was developed as part of the development of modern telecommunication facilities and has impacted the measurement of runtime complexity decisively. *Kolmogorov complexity* is a measure for the shortest program that can output a given string [CARDOSO 2006]; it plays an important role in the formalization of software code and has, as such, laid the foundation for the formulation of effort calculation as shown above, for example, in Halstead's metrics.

Table 2-12: Common structural metrics for object-oriented programming paradigms

Metric	Description (for details, see [WAND & WEBER 1990])
<i>Methods per class</i>	The metric methods per class sum of all method-related metrics (all others) to estimate the overall complexity of a program.
<i>Depth of inheritance tree</i>	The depth of the hierarchy measures which attributes are inherited from the original class and estimates the impact and propagation of possible changes (i.e., how many other classes can be reached by a change).
<i>Number of children</i>	The number of successors of a class estimates the impact that changes within the program have on a particular class (i.e., the reachability).
<i>Coupling between object classes</i>	This metric enumerates the number of other classes to which a class is related to estimate errors and clear the structure of a program.
<i>Response for class</i>	This metric defines the size of the set of methods of a class that can respond to a specific message to estimate the run-time complexity of a program.
<i>Lack of cohesion</i>	This metric provides a comparison of variables and methods, and estimates thereby the necessity to split a class into two or more classes.

In summary, metrics from software management can be classed as highly relevant for process analysis, as software is similarly characterized by many decisions that cannot be analyzed in a deterministic manner. Software engineering offers several metrics, including empirical foundations, because of their relevance in software

design. The transfer of these foundations to process management is highly possible, as the following sections will show.

2.3.4 Metrics in process management

Structural metrics for process assessment are a recent development [GHANI et al. 2008]; while quantitative measures have been used for a long time, the pioneer work on qualitative metrics occurred only in the 1990's with the assessment of Petri nets of the first structural and dynamic metrics [LEE & YOON 1992].

Today, a number of metrics exist⁵². Yet, these metrics have neither been consolidated nor compiled into a coherent body of knowledge; so far, the knowledge on qualitative assessment of processes and their structure is still fragmented [MENDLING 2008, p. 114].

Generally, two kinds of metrics are in use: those that are mainly used for the prediction of the behavior of a process, and those that are used to estimate errors in a process model [GRONBACK 2006] [GRUHN & LAUE 2006a]. However, the border between the two kinds cannot be clearly defined, as often process models are designed in a way that is not completely free of errors, while deviations from a semantically and semiotically correct model are intended to transmit a certain purpose or meaning [MENDLING et al. 2007]. [Table 2-13](#), therefore, does not differentiate the two kinds.

Overall, many metrics are similar and use comparable concepts that have been, in part, embodied independently from each other. At the same time, many approaches remain conceptual and without empirical evidence. Lastly, few authors provide detailed algorithms that can be used to compute⁵³ the metrics.

Metrics for business processes represent thus the main contribution that is used to answer the research question behind this book. Although many were developed for business processes and workflows, they can be transferred to engineering design processes without limit, as engineering design processes are basically a subclass of a business process, being more complex and less deterministic in their overall behavior (compare [Table 2-6](#)).

⁵² A detailed overview of metrics with a structural focus on workflows and business processes is provided in [MENDLING 2008, pp. 110-117]; the earlier work is also comprehensively summed up in [CARDOSO et al. 2006], and [GRUHN & LAUE 2006a].

⁵³ As complexity metrics are difficult to compute, tool support is necessary. Currently, there are three software tools available that embody a wide set of metrics: KOPeR [NISSEN 1998], EPCMetrics [GRUHN ET AL. 2006], and STAN [BECK & STUHR 2008].

Table 2-13: Overview of common metrics in process management (based in part on [MENDLING 2008])

Metrics	Description
<i>Distinct paths, hierarchy levels, cycles, diameter, parallelism</i> [NISSEN 1998]	As one of the first concepts for metrics to describe a process, Nissen's metrics are meant to facilitate the (re)design of a process. Nissen counts the number of independent paths, the levels of hierarchy found in the model, the number of independent cycles, the overall diameter (i.e., the longest path), and the degree of parallelism among activities.
<i>Simplicity, flexibility, integration, efficiency</i> [TJADEN et al. 1996]	Tjaden et al. developed their approach focusing on establishing a balance between the four metrics they designed. Simplicity represents basically the size of the process; the flexibility and integration level are not actually calculated but scored using a checklist-like function point approach. Tjaden et al., however, do not provide a detailed concept of efficiency. Therefore, their approach is criticized as too abstract [BALASUBRAMANIAN & GUPTA 2005].
<i>Size, length, structural complexity, coupling (concurrent and sequential contribution, arcs of a subnet)</i> [MORASCA 1999]	Morasca bases his metrics on a detailed, formalized theoretical foundation to assess Petri-net process models. As size, he uses the number of places and transition, as length the diameter of the network, as structural complexity the number of distinct paths, as concurrent contribution a count of edges, as sequential contribution a count of places, and finally an assessment of modules and their incident and outgoing edges as a measure for modularity. However, his approach is purely theoretical and not empirically supported.
<i>Network complexity, Cyclomatic number, reduction coefficient, restrictiveness, number of trees</i> [LATVA-KOIVISTO 2001]	Latva-Koivisto uses established foundations to build a set of metrics that describe a process in a complete manner; however, his approach is criticized as incomplete, as it does not differentiate different entities in the process model, and it lacks more tree-related metrics [MENDLING 2008, pp. 115]. The network complexity is similar to the density of a network, the cyclomatic number is similar to that by [McCABE 1976], the reduction coefficient calculates the number of steps to reduce the network to a maximum path length of 1, the restrictiveness estimates the number of possible sequences in a process, and the number of trees assesses the existence of hierarchies that permeate the process.
<i>Cohesion of functions, events, and data objects</i> [DANEVA et al. 1996]	As a toolkit for EPC, the cohesion of functions among each other represents the intensity of involvement of a function in the control-flow; events and data objects in the control-flow are assessed similarly. The authors conclude that their measurement suite is mainly suitable to identify error-prone fragments of the process.
<i>Cycle Sets (number of cycles, entry nodes into cycle, exit nodes out of cycle), structuredness (existence of deadlocks), nesting depth</i> [GRUHN et al. 2006]	The authors extend, essentially, the count of cycles as in [NISSEN 1998]: They integrate the count to those nodes that serve as an entry point or an exit to cycles. Equally, they assess process models for the existence of deadlocks (which later is completed by [MENDLING 2008]), i.e., joins and splits that are not harmonized. Ultimately, they introduce a measure for the nesting depth, i.e., the maximum number of splits in a given path across the process.

Table 2-13: Overview of common metrics in process management (continued)

Metrics	Description
<i>Control-flow complexity</i> [CARDOSO 2005a]	The control-flow complexity is an adaptation of McCabe's Cyclomatic number, integrating the characteristics of how the process can continue after each decision point that splits the control-flow. Well validated, it is not designed to predict errors.
<i>Log-based complexity</i> [CARDOSO 2007]	Log-based complexity extends Cardoso's control-flow complexity, having a structural focus, to a run-time measure that assesses the process complexity based on the log data of how the process is executed; it is, thus, only suitable for formalized workflows.
<i>Maintainability (analyzability, understandability, modifiability, all using the size, complexity, coupling of a process)</i> [ROLÓN et al. 2006a]	Rolón et al. estimate the maintainability of a process, breaking it down into three concepts: The analyzability addresses the probability of discovering errors in the process, the understandability estimates the ease of comprehension of the model, and the modifiability is a measure for the probability of making the correct changes to erroneous models. The metrics are based on size (counting all entities in the process), complexity (based on the number of dependencies), and coupling (between activities).
<i>Degree of automatization, activity automation, role integration, role dependency, activity parallelism, delay risk</i> [BALASUBRAMANIAN & GUPTA 2005]	This set of metrics supports process planning by supporting the evaluation of different alternatives for a given process. To this end, they extend the existing pool of metrics by a few more metrics that address the possibility of automated process execution and decision making where possible, the purposeful integration of resources, the measurement of the degree of parallelism among tasks, and the identification of structural bottlenecks that possibly cause delays.
<i>Cognitive weights</i> [GRUHN & LAUE 2007a]	The key idea of their approach, being based on [SHAO & WANG 2003], is to assign a cognitive weight to basic software control structures: the more difficult a control structure is to understand, the greater its cognitive weight. The approach is based on empirically funded coefficients for basic patterns in the control-flow.
<i>Cognitive cross-connectivity</i> [VANDERFEESTEN et al. 2008]	The proposed metric measures the strength of the links between pairs of entities in the process based on the hypothesis that encompassing "the structural relationship between any pair of model elements" is a complex mental operation.
<i>Understandability</i> [GHANI et al. 2008]	Besides other common complexity measures, Ghani et al. introduce the understandability of a process model as a contribution to its maintainability. They propose calculating it based on anti-patterns, for which, however, they do not provide empirical evidence.
<i>Size, density, partitionability, connector interplay, cyclicity, concurrency</i> [MENDLING 2008, pp. 117-130]	Mendling collects and extends metrics and their empirical foundations to verify process models and predict possible errors. These metrics make use of the strong formalism that guides the EPC process modeling language, and the different metrics embody these formalisms to assess EPC models for their formal correctness.

2.3.5 Metrics for engineering design processes

Generally, metrics in engineering design processes are meant to serve three purposes: estimation, monitoring, and performance measurement. Yet, there is generally little specific work on metrics for engineering design processes available⁵⁴. This is mainly because product development has the nature of “a mental exercise” and because of “a lack of easily identifiable items to measure” [BASHIR 1999]. It is true that the existing metrics, therefore, remain either highly specialized, or they are conceptual and hard to apply.

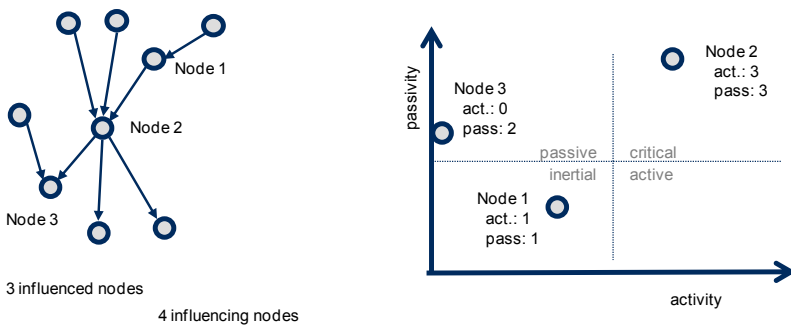


Figure 2-24: Activity and passivity of the elements of a system

The most common metrics⁵⁵ are very simple. Having originated from systems engineering, the measures of *activity* (also called active sum) and *passivity* have a strong structural focus. They compare the immediate impact of neighboring entities on each other and thus are similar to the degree-measure that is a common basis in graph theory [DAENZER & HUBER 2002, pp. 558-560] [LINDEMANN 2007, pp. 73-76].

Other metrics in engineering design incorporate the numerical evaluation of the product or process architecture to some extent. Mostly, these metrics are designed as effectivity and efficiency measures, for example, for attributing the necessary manpower [NORDEN 1964] or for estimating the *degree of efficiency* of a design process [O'DONNELL & DUFFY 2005, pp. 70-79]. Similarly, there are metrics to assess the impact of project characteristics on process planning [CLARK 1989]. These involve, for example, the level of risk in new product development, which

⁵⁴ [BASHIR 1999] provides a sound overview of the state of the art of metrics in engineering design at the time of writing, while [HORNBY 2007] provides a recent overview of common and more specific measures (which are not further regarded here).

⁵⁵ Score evaluations and similar methods are common in engineering design, e.g., in risk management [LINDEMANN 2007, p. 276]. As they have no explicit focus on structure, they are not considered here.

is broken down into metrics for *product innovation*, *product complexity*, *design maturity*, and *schedule pressure* [ZURN 1991]. Also, lead time can be broken down into the driver's *product complexity*, *management complexity*, and *amount of charge* [GRIFFIN 1993]. LIU ET AL. establish a metrics-based process review from a structural point of view [LIU et al. 2003]. They apply theoretical measures to assess the *critical degree* (a weighted measure of the task precedence in the process), the *likelihood of error occurrence* (based on an estimation of the novelty of a product and the availability of knowledge in the company), and the *spread degree of a task* (a weighted measure of the reachability of subsequent tasks) to support the planning of a process based on other process reviews.

As can be seen from these few process-based approaches, the estimation of the design complexity⁵⁶ (i.e., a measure of how complex a product is) is the focus in all approaches, as the product complexity necessarily drives the process complexity. Early works measure design complexity by the *coupling between the design targets and their variables* [DIXON et al. 1988]. SUH develops different *metrics for function coupling* [SUH 1999], and other authors introduce *estimators of design complexity* [BASHIR & THOMSON 1999] [SUMMERS & SHAH 2003]. HORNBY ultimately introduces *modularity* (based on the number of modules and the degree of their interaction with the overall system), *reuse* (measuring the repeated occurrence of entities within the design) and *hierarchy* (similar to the nesting depth in process management) as classifying measures for product complexity [HORNBY 2007].

A measure that focuses purely on process complexity is illustrated by [SCHLICK et al. 2008]. Here, a numerical DSM is used to model project dynamics; the approach is able to cope with large teams “who make at least partially autonomous decisions on product components but also strongly interact in their impact on project performance”.

There are, of course, many other measures available that, however, do not relate to structural complexity but are measures used in, for example, benchmarking projects, process audits, or performance measurement for management and organizations (mostly financial and operational measures in project management [PMI 2003]).

In summary, although engineering design science strongly focuses on model building, there is virtually no work on complexity metrics available [BENSON 2007], even less so for processes. This can be attributed in part to the fact that engineering design processes can be treated like business processes concerning their analysis; however, the specific interpretation basis for such processes cannot be directly transferred, and there is a large gap in science about the meaning of the available structural metrics for processes.

⁵⁶ Compare [AMERI et al. 2008] for a detailed comparison of existing measures of product complexity.

2.3.6 The limits of using metrics in an organization

GRIFFIN points out that measurement is an important first step towards process improvement [GRIFFIN 1993], as it is important to have a sound overview of the initial state, the needs and development potential, and a final comparison after improvement measures have been implemented.

Yet, metrics are no remedy for any problem. “What you measure is what you get” was the driving dictum for the research of the Balanced Scorecard, at the time revolutionary, as it allowed a wider, balanced picture that substituted a former management that was only guided by financial goals (see section 6.1.3 for more details). To this end, metrics already show their biggest disadvantage: While reducing the complexity of the representation of an issue, they tend to oversimplify or omit dependencies of an issue, thus making the representation incomplete [KAPLAN & NORTON 1992]. It is, therefore, necessary to select a group of metrics to represent a problem in a balanced way.

Secondly, metrics directly impact the behavior of personnel in a company. In particular, in the management concept “Management by Objectives”, metrics are a common basis for the evaluation of the performance of personnel [DRUCKER 2007, p. 261]. The concept is based on goals for each member of a company, whose fulfillment is measured to assess the individual performance [ODIORNE 1980, p. 82]. As part of this measurement, the motivation of an employee is directly related to the results of the measurement; in turn, a measurement influences the behavior of an employee in a positive way, but it can equally demotivate [MUTSCHELLER 1996, p. 61]. Thus, measures need to be chosen carefully to ensure they are not influenced by staff out of fear or personal ambition. This leads to two consequences: One the one hand, staff needs to be integrated early into the process of installing measures in the company to achieve a transparent measurement system that is favorably accepted and thus maintained; on the other hand, metrics need to be changed at regular intervals to ensure that the staff also considers other relevant specifics of a situation that are not part of the measures in place.

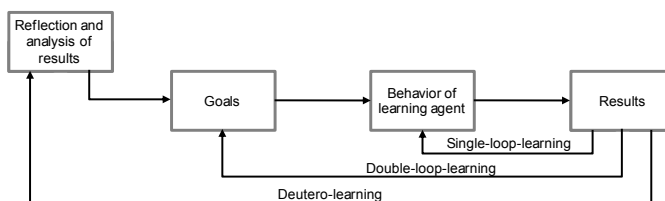


Figure 2-25: Three types of organizational learning

Organizational learning similarly stresses the fact that “maps and images” guide the behavior of individuals in an organizational setting [ARGYRIS & SCHÖN 1978, pp. 17-19]; metrics can serve as such images that are considered attainable and therefore guide individuals without considering other aspects. In this context, individuals take on the role of a so-called learning agent, whose learned behavior

is guided by the impact of their actions [ARGYRIS & SCHÖN 1978, p. 29]. Three types of learning are common, which involve more reflection of the third type, the deuteron-learning (see Figure 2-25). Generally, single- and double-loop learning is guided by the results of the behavior, upon which measurement of the results can have a detrimental influence if metrics are designed in a way that neither stipulates reflection nor is set up in a holistic way to avoid one-sided behavior.

In engineering practice, metrics are mostly used for project controlling; they are rarely used for concept and test design [FINK & HAMPP 2005]. An empirical study in the software industry revealed five common strategies as to how engineers commonly cope with metrics in a company [SIMON & SIMON 2005]:

- Optimism-strategy: Personnel disapprove of the metrics, perceiving them as implicit criticism and a constraint on their professionalism.
- Delegation-strategy: The results of metrics are attributed to external reasons that are not related to an individual engineer's work, and thus responsibility is denied.
- Automatism-strategy: Problems that surface through the metrics are blamed on tool-support (automated workflows, or, in software engineering, code generators).
- Particularity-strategy: As design in problem solving, metrics are not recognized as relevant to the specific issue, and their validity for common situations is denied.
- Tortoise-and-Hare-strategy: Refers to the fact that an issue was already improved before metrics were introduced, and engineers tend to turn away from a problem.

Generally, it can be stated that metrics provoke a lack of emphasis on the environment in terms of a goal (and its respective measure) due to a lack of understanding of the actual system [DEMING 1994]. Three “traps” are inherent in such measurements:

- Common and special situations are little differentiated by such measurements, although criteria are not universally valid. Typically, measures depend on other external influences; deflections thus must exist.
- A single measure is not the best criterion to judge an issue, and often the focus is misplaced. A more comprehensive set of metrics helps avoid single-sided improvements. The overemphasis of individually measured aspects can, in some cases, lead to “bogus metrics”, i.e., metrics that mislead the company [BOLLINGER 1995].
- The common assumption that improving a single measure improves the overall system is wrong, as in most cases a more holistic view is needed to correctly assess a system.

It is possible that a set of metrics is misleading in the long term. It is generally recommended that measures be alternated from time to time and there be overlapping measures that exercise a certain control over each other, which, at the same time, lowers the risk of manipulation of an individual measure [KAPLAN &

NORTON 1992]. The goal is to lower the risk of simply seeing the measure and not seeing the object behind the measure. Furthermore, a balanced set of several measures works best to achieve a comprehensive picture.

Metrics for analysis should, therefore, not be applied in isolation, as they do not describe “goodness” per se. Rather, they point to possible quality issues that need to be further evaluated [BOLLINGER 1995], which in this research is targeted by assessing outliers to show how a process improvement project can be prioritized and where possible improvements can be expected.

2.3.7 Summary

As the most reduced model possible, metrics are able to represent a system in a condensed form to show important characteristics and to point to important aspects. In process analysis especially, metrics are a tool for the identification of weak spots and their conditions [BENSON 2007]. As such, metrics can introduce the risk of reduction past a meaningful limit (reductionism).

At the same time, metrics need to be used carefully in a company, as they necessarily influence the behavior of staff, especially in the context of management by objectives, where objectives are related to measures. They should, therefore, be used mainly to focus on further investigations [BENSON 2007].

There is a substantial body of metrics available that is able to assess the structural complexity of a system with a view to different patterns. These metrics are used, on the one hand, to discover modeling errors, and, on the other hand, to better understand a system’s behavior through the measurement. Many different metrics for the analysis of network structures in all kinds of disciplines exist and can be transferred to process management; yet no comprehensive compilation is available. At the same time, the transfer to the specifics of engineering design processes, i.e., what behavioral aspects relate to what structural characteristic evaluated in a metric, remains unsolved. There is, especially, no systematic listing of the significance of the available metrics against the domains and relationship types common to process management.

Commonly, metrics are not independent of each other but can be organized in a measurement system (according to, for example, focus, goal, granularity). This enables the systematic and goal-oriented employment of metrics. This is especially important in regards to the structural analysis of a process, as a metric can only be purposeful in the context of a goal and the related semantics; metrics, therefore, cannot be designed without a meta-model that provides a semantic context to later interpret the metric.

2.4 Directions from the state of the art

The management of engineering design processes is a wide field of research; there are many different models and methods available, few of which, however, are designed to cope with the structure of a process in a comprehensive manner. Nevertheless, the management of structural complexity and related fields of science have provided different means of understanding, modeling, and managing relationships in a complex system.

Both process management and the different sciences related to structural complexity have created a number of different metrics that evaluate the complexity of a system based on various structural characteristics; many of them are empirically validated and recognized in research and application, and the transferability to the management of processes is generally confirmed. [Table 2-14](#) summarizes existing metrics and the structural criteria they assess: A mark in a cell relates a metric and a structural characteristic that the metric focuses on; however, it is possible that a metric also includes other structural characteristics that are not essential to its function. The table was generated by reviewing every metric (as shown on page 80 and the following pages) for its basic structural focus, as described in the literature.

As the table shows, some structural characteristics are not evaluated yet; however, a substantial toolset exists that is suited for the structural analysis of a process. In particular, n-partite-ness, isolated nodes, leafs, bridge nodes, biconnected components, spanning trees, the Small World effect, transitivity, degree distributions, navigation, and centrality are not evaluated as recognized metrics, although some of these structural characteristics have, in fact, the character of metrics themselves. These need to be reviewed in detail for their use in extending the set of means of analysis for processes.

Furthermore, process management and the analysis of processes is generally a goal-oriented procedure, which works to improve a process for one or another concept. As a review of the literature shows, the common goals of process management (planning, resource consumption, quality, flexibility, organizational decomposition, interfaces, and transparency of process, see page 66 and the following pages) and the typical properties of design processes (dynamics, creative nature, loops, leaps, iterations, results that are not predictable, changes, imperfect definitions, uncertainty and risk, maturity of artifacts, process path not determinable, and the involvement of many stakeholders, see page 61 and the following) and the means of analyzing a structure have not been systematically used in conjunction, and, therefore, no mapping between them is available. However, as individual works show, the goals can be related to certain structural patterns via the individual process patterns; for example, the triangularization of a DSM of tasks relates directly to the reduction of leaps and loops by proposing an improved sequence of tasks, thus contributing to better process planning.

These structural characteristics may possibly occur for all domains and relationship types that exist in process models. In fact, it is necessary to know the specific semantics of a process model to give meaning to the structural characteristics and to the structural metrics, as only the semantics of the nodes and edges of the underlying network structure allow indications about the behavior to

be deduced. However, the specific attribution of all structural characteristics to all relevant domains has yet not been researched. Thus, no dedicated means of systematical analysis exists, but only fragmented parts thereof.

As common domains, tasks, artifacts, events, organizational units, resources, and time, were identified as basic domains of process management ([Table 2-9](#)). These can serve as an approximation to interpret the structural metrics and give them significance, which, however, will be precise if the relationship type is also considered in the second step. However, in many cases, this procedure will be too complex to be handled. To facilitate this procedure, principal relationship types were identified.

Complexity Metrics in Engineering Design
Managing the Structure of Design Processes

Kreimeyer, M.; Lindemann, U.

2011, XIII, 403 p. With online files/update., Hardcover

ISBN: 978-3-642-20962-8