**INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'09**
24 - 27 AUGUST 2009, STANFORD UNIVERSITY, STANFORD, CA, USA

# A PLANARITY-BASED COMPLEXITY METRIC

**Sebastian Kortler, Matthias Kreimeyer, Udo Lindemann**
Institute of Product Development, Technische Universität München

## ABSTRACT

Complexity in product design, e.g. in product architectures or process structures, continuously increases. To properly evaluate a complex system or to compare it to other systems, complexity metrics enable the numerical assessment of a system's underlying structure. The more complex such a structure is, the more likely it is to have many linkages between the elements. Typically, the number of edges that cross each other increases with the degree of complexity. This fact is used to derive a complexity metric that computes the distance of a given network from an ideally planar graph, i.e. a network that has no edges that cross each other.
The approach proposed closes an existing lack of structural criteria and metrics and complements the existing possibilities to measure structural complexity, i.e. the particular interaction of a system's elements and their interdependencies. To this day, planarity is not used in this context.
The metric is developed based on a state-of-the-art algorithm and tested with a small example from an industrial design process to illustrate its independence to existing complexity metrics.

*Keywords: complexity, planarity, metric, structural criteria, systematic assessment, process metric*

## 1 INTRODUCTION

Manufacturing technical products implies complex design processes. While there are many facets to such complexity, one perspective is to characterize products and their design processes by their underlying structures. In order to handle and manage such structures, various methods e.g. from systems engineering can be used. However, comparing and evaluating the characteristics of a complex structure makes it necessary to measure the degree of complexity by determining underlying patterns and different structural criteria and then evaluating their extent and impact numerically. To do so, complexity metrics provide the possibility to compare and to assess such structures.

### 1.1 Complexity in product design – an industrial case

Complexity in product design continuously increases. In order to pass the challenge of growing complexity caused by markets, products and processes, companies have to control their design processes and the underlying structure, as this structure governs the system's behavior [1].

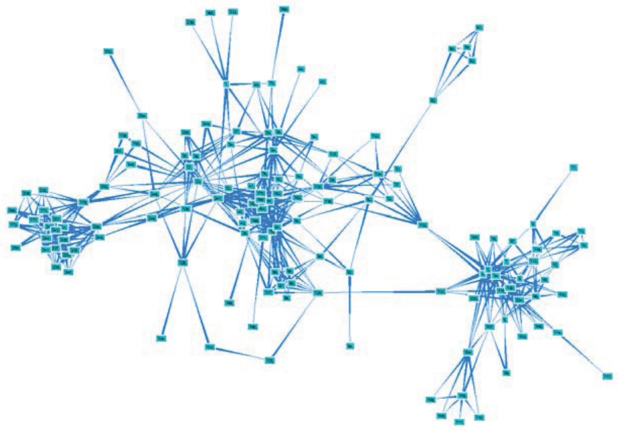

*Figure 1: Part of the dependencies between tasks in an automotive design process*

Complexity in a product can result for example from variants; considering e.g. the development of car components, different variants result from the progress and adaptation of certain parts. Typically, not all of these variants can be combined with each other. Changing one single component may therefore result in extensive impacts on other components, causing high costs and many secondary adaptations [2]. An evaluation of change propagation complexity levels could therefore be an efficient means for management to assess on – a strategic level – what risks might be implied changing the product architecture at what level.

Figure 1 shows a snapshot from a graph representing how tasks of a design process are interlinked via the documents that are produced by each task or that serve as an input to a task: Each node (numbered for nondisclosure reasons) represents a task, each edge linking two nodes at a time represents a document. The figure vizualizes a section of an automotive design process for a premium class SUV, in particular the design, simulation and testing of onboard electronic equipment (mainly controller design). In total, the process consists of 15 sub-processes (pre-defined modules within the overall process, three of which can be seen in the figure), set up from 377 tasks and 237 different documents. In average, each task has 1.65 edges, i.e. the process is rather densely networked, especially considering the fact that the process model is only of average quality, as 51 tasks are unconnected.

As the process is broken down into 15 different sub-processes, one relevant question for process management is how these sub-processes can be prioritized when it comes to re-engineering or improving them. To this end, among other aspects (e.g. run-time, total cost, available human resources,…), the structure of the process gives evidence how easily the process is to be understood by an engineer involved in the process, and the evolution of the structure over time can point to the process that has grown the most complex and that needs attention first. Therefore, a metric that represents the degree of complexity to the system "process" could serve as a means of better assessing such a system.

In summary, complexity appears to be many-sided, being situated everywhere. However, different systems can have different levels of complexity; a simple device can be little complex, while an aircraft certainly will be highly complex. Nevertheless, the underlying structure follows the same basic principles [3, 4] at a more abstract level. For many applications it can be of interest to assess the degree of complexity of a system (or a specific part or view thereof), especially when it comes to comparing different systems of the same kind.

## 1.2 Problem description

The most common way to measure a system's quality is by applying metrics that characterize the system properly. There have been different approaches to measure the complexity of a system (cf. section two). Yet, as will be shown, no approaches have been made to evaluate the degree of planarity of a network structure representing a system: The more complex a system, the more likely it is to have many linkages between the elements; the more linkages there are, the more likely it is that these linkages cross each other, which in graph theory refers to the fact that the structure is not planar. The concept that is thus presented in this paper is to characterize the degree of complexity of a system by its measure of its distance from an ideal plane wherein the structure would be fully planar.

## 1.3 Definitions

***System***

A system is created by entities and their interdependencies that form a system's structure, that possess individual properties, and that contribute to fulfill the system's purpose [5]. Systems are delimitated 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. In this paper variation over time is not considered [6].

***Complexity***

*Complexity in product development does not represent a problem per se; rather it is the lack of ability of users to control complexity that leads to severe consequences. Consequently, improved control of complexity allows for enhanced possibilities in product development* [3]. In fact, this definition relates to the fact that a system is perceived as complex (in contrast to "simple"), if it is difficult to understand and verify because of its large number of elements, dependencies, and states.

***Structure***
“Structure” is understood as the network formed by dependencies (edges) between a system’s entities (nodes). It furthermore relates to the semantics of this network; the structure of a system therefore always contributes – in its constellation – to the purpose of the system. Structures and their subsets can be analyzed by means of computational approaches, primarily provided by the graph theory and related sciences [3].

***Structural Characteristic***
A structural characteristic is understood as a particular constellation of nodes and edges, i.e. it is formed by a particular pattern formed from nodes and edges ([3][7]). 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 [5]. A structural characteristic only possesses significance in the context of the system it is describing.

### 1.4 Structure of this paper
Having elicited the problem this paper focuses on and the required basic terminology, first, a state of the art is laid out to explain the background of how structural complexity can be managed; in addition to this, a differentiation of structural characteristics and existing complexity metrics is presented and classified against its mathematical foundations to discover possible gaps in the research and application of methods of structural complexity (section 2). To fill the gap that is identified for planarity, a planarity-based metric is developed using the state-of-the-art algorithms available (section 3). The metric is tested on four real examples to prove its validity (section 4). Section 5 concludes the paper with a short reflection.

## 2 RELATED WORK

### 2.1 Structural Complexity Management
Most commonly, structural complexity management is based on matrix-based approaches that represent how elements (as rows and columns) are interconnected (by a number in the cell that links a row to a column). Originating from a process focus with the first published formulation of a Design Structure Matrix (DSM) [8], a whole community has developed around this research. The DSM is able to model and analyze dependencies of one single type within one single domain. Browning [9] classifies four types of DSMs to model different types of problems: component-, team, activity-, and parameter-based DSMs. However, many other classifications exist (e.g. [3]) nowadays.
There are numerous algorithms to analyze the overall structure of the relationships within a DSM; starting from the original algorithms for tearing, banding and partitioning [10][11] to a still non-exhaustive list provided by [3].
The authors of [12] have extended DSM to DMM, i.e. Domain Mapping Matrices, to enable matrix methodology to include not just one domain at a time but to allow for the mapping between two domains. [3] has taken this approach further to model whole systems consisting of multiple domains, each having multiple elements, connected by various relationship types. The author refers to this approach as Multiple-Domain Matrix (MDM), providing a number of ways to analyze the system’s structure across multiple domains, condensing each single analysis into one aggregate DSM.
In fact, many of the algorithms that are used in matrix-based methodologies are applications of graph theory, which provides for the mathematical foundation of describing networks. To this end, network and graph theory are closely interconnected and not easy to separate. Whereas network theory focuses on the global features of mostly random networks, graph theory addresses structural features that originate from the interaction of single nodes and edges of a network structure. Graph theory is often traced back to Euler’s works (e.g. [13]), and network theory can be comprehended as the research of [14-18].

***Classification of structural characteristics***
Almost all of the approaches to structural complexity management look into what characteristic qualities can be found in a structure, from the level of a global structure down to the integration of individual nodes. However, structural characteristics and complexity metrics are rarely discerned

strictly. While a structural characteristic relates to the pattern of nodes and edges as shown in figure 2, a complexity metric relates to condensing one or more structural characteristics into a numerical value. Figure 2 orders the structural characteristics as provided by [3] by the evaluation of the number of edges and nodes that form a structure. In fact, most of the characteristics can be traced back to a few basic elements (e.g. a hierarchy is a special kind of path taking attainability into account).

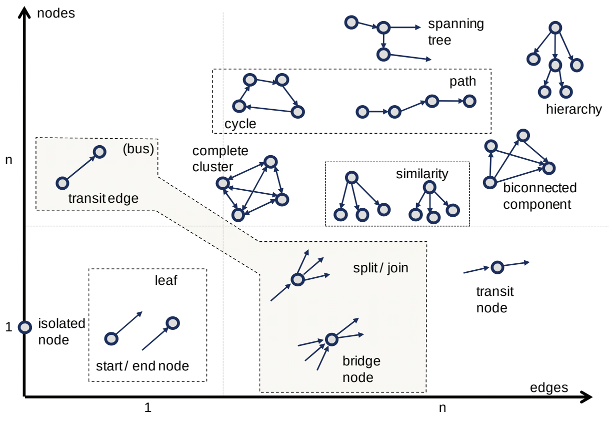

*Figure 2. Basic structural entities*

## 2.2 Complexity Metrics

***Basic complexity metrics***

Complexity metrics are most common in software engineering. There, the computational complexity rather refers to the degree of computational power that is needed to execute an algorithm [19]. However, there is also much work available in software engineering that focuses on quality metrics for software code. [20] gives a comprehensive overview of how to use these metrics. The focus is on using metrics for the understanding, evaluating, controlling and forecasting of a software program. Most of the metrics focus, in fact, on evaluating the dependencies within a software code, and as such are closely related to structural complexity. A set of requirements to evaluate such metrics is know commonly as Weyuker's properties exists to evaluate whether a metric is a good metric [21].

Other researchers built on this foundation and set up metrics to describe workflows, mostly based on the EPC notation, evaluating them for possible bottle necks, modeling errors and the robustness of the overall structure of the workflow concerning decision points [7, 22]; again, the authors use Weyuker's properties to evaluate their metrics [23].

[24] develops a general measure of structural complexity, regarding the similarities and differences between design problem complexity, design process complexity, and design artifact complexity. The authors indentify size, degree of coupling, and solvability as the three fundamental aspects of complexity, and measures for each aspect are defined. However, these measures are developed specifically for parametric and geometric problems.

Based on a stochastic model, [25] develops a complexity measure that estimates the complexity of a process by evaluating the interaction between partially autonomous actors in a concurrent engineering setting. The measure evaluates numerical DSMs, i.e. it is based on weighted edges, and estimates the amount of information necessary to complete a process.

A more pragmatic set of metrics is proposed by [26]. In this approach, different structural criteria from [3] are condensed into 53 different metrics that are used in conjunction with a process meta-model to describe the different aspects of a process as comprehensively as possible.

***Complexity metrics involving planarity***

To assess a network's understandability [27] used planarity, as it allows for the determination of the ascertainability of the network model. There, the lowest number of dimensions to obtain a planar graph serves as a cognitive volume in order to describe the transparency of the process model.

[28] develops a cognitive weight to evaluate software complexity by examining the cognitive weights of basic control structures of software. As such, it is based on empirically founded characteristic values. The measure is used as a description of the human ability to grasp the global structure being made up of individual parts. The work is based on the hypothesis that the more "intertwined" a structure of a flow is, the more complex the system is. To measure this degree of complexity, the structure is broken down into individual nodes and their immediate surroundings, to which then an individual cognitive weight is attributed. The approach thus does not describe the degree of planarity directly, but it only evaluates each node's individual contribution to a possibly non-planar graph.

***Complexity metrics involving planarity***

Structural characteristics and structural metrics sometimes overlap, while all make use of phenomena that are described in graph theory. Table 1 illustrates the available basic phenomena in graph theory, based on [13]. Although there is no complete one-on-one relationship between phenomena, structural criteria and structural metrics possible, the table regroups what phenomenon a structural criterion focuses on; the same is done for complexity metrics. 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). While there has been some work into planarity so far, no detailed analysis of its general applicability as a structural criterion that can be found (meaningfully) in any kind of structure has been made to the best knowledge of the authors. Having proven basically useful, nevertheless, as shown in the previous section, this research therefore further investigates the usability of planarity as a measure for complexity to extend the available set of means to describe structural complexity.

*Table 1. Phenomena in graph theory and their application in structural criteria and complexity metrics*

| Graph Theory | Structural Criteria available | Complexity Metrics available |
|---|---|---|
| Cliques, Subgraph | Strongly Connected Components (√) | √ |
| Walks | Cycles (√)<br>Paths (√)<br>Distance (√) | √<br>√<br>√ |
| Trees | Leafs (√)<br>Roots (√)<br>Spanning Trees (√)<br>Knots(√) | √<br>√<br>√<br>√ |
| Minors, Embeddings | ? | |
| Adjacency and Degree | Neighborhood (√)<br>Bridges(√)<br>Degree (Activity…) (√)<br>Independence (×)<br>Connectivity (Attainability,…) (√) | ×<br>√<br>√<br>√<br>√ |
| Genus | Planarity (×)<br>Thickness (√) | (√)<br>(√) |
| Weighted graphs and networks | Weighted Nodes (√)<br>Weighted Edges (√)<br>Minimum Spanning Tree( √)<br>Shortest Path (√) | √<br>√<br>√<br>√ |
| Coloring | Chromatic Number (√×)<br>K-Coloring (×)<br>Color-Classes (×) | ×<br>×<br>× |
| Multipartite Graphs | Disjunctive sets (×)<br>N-Partite Graphs (√) | ×<br>√ |
| Eigenvalues | Eigenspectra (×) | × |

## 3 PLANARITY AS A COMPLEXITY MEASURE

### 3.1 Planarity

A planar graph is defined as a graph that **can** be embedded in the plane with all its edges and nodes not crossing each other. Figure 3 gives an example. According to Kuratowskis's Theorem, a finite graph is planar if it does not contain a subgraph that is a subdivision of $K_5$ or $K_{3,3}$ (because the graphs the $K_{3,3}$ and the $K_5$ represent the smallest non-planar graphs, see figure 4) [29].

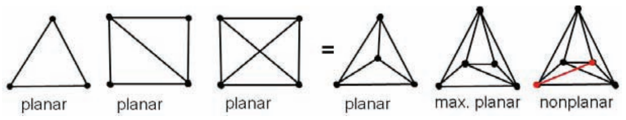

*Figure 3. Planar and non planar examples*

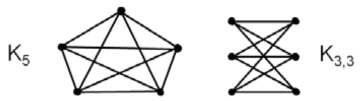

*Figure 4. The smallest non-planar graphs*

***Planarity Testing - available algorithms***

Testing for planarity is a rather difficult problem, for which a number of algorithms have been developed. *Path addition* [30] was the first planarity testing algorithm working in linear time. It uses depth-first search and implements an iterative version. *PQ tree vertex addition method* [31] is based on the PQ tree data structure (i.e. a tree-based data structure that represents a family of permutations on a set of elements). It embeds one vertex of a graph at each step in an order given by a "st-numbering" of the vertices (i.e. a particular way of labeling each edge, in which neighboring edges are numbered in a way that each edge has a neighbor with one lower and higher number as a neighbor). After each step permuting and reversing pieces of the graph are used to preserve planarity. Later, these two methods were combined to form the *PC tree vertex addition method* [32]. This planarity testing algorithm is significantly simpler than classical methods based on a new type of data structure called the PC tree and a postorder traversal of the depth-first search tree of the vertices. The first simplified O(n) (i.e. allowing for computation in linear time) algorithm was the *edge addition method* [33], originally inspired by the PQ tree method. Instead of PQ trees it uses an edge to be the fundamental unit of addition to the partial embedding, while preserving planarity. The algorithm uses the fact that subgraphs can become biconnected by adding a single edge. This method is currently state of the art and therefore used in this work.

### 3.2 Strategies to describe the degree of planarity

Determining if a graph is planar is relatively easy for a small graph. However, the more the degree of cross-linking rises, the lower the probability of planarity. Measuring complexity in product development implies working with systems which elements are interlinked profoundly. Thus, the basic planarity criterion is not applicable unless extended to work with large structures.

In order to describe how far away from planarity a given graph G = (V, E) is (V is the set of vertices or nodes, E the set of edges), the degree of planarity is used in the maximal planarization problem (maximum planar subgraph [34]). Maximal planarization generates a subgraph G' of G, to which no edge of G – G' can be added while preserving planarity. A maximum planar subgraph G' is a subgraph of maximum size amongst all planar subgraphs of G. In order to generate the maximum planar subgraph, different maximal planar subgraphs need to be compared with each other.

The maximum planarization problem is NP-hard (i.e. it is a nondeterministic problem that can be computed in polynomial-time). As such, the maximum planarization algorithm provides a measurement which quantifies the distance between the respective structure and a planar structure in one special case. Hence, the measurement is not qualified to serve as a complexity metric. In order to use the degree of planarity, it is necessary not only to evaluate special cases of the given structure, but all cases. Therefore, the measurement that is proposed here uses the mean value across all distances of all maximal planar graphs from the original graph as a complexity measure.

## 3.3 A metric describing planarity (description of generating a mean value)

The basic idea to measure complexity in structure using planarity is to count the number of edges, which need to be removed for preserving planarity (maximal planarization). However, the number of edges which need to be removed depends on the ordering of the edges as shown in Figure 5. Choosing the minimum number of edges, which need to be removed, will lead to the maximum planarization problem. Considering the ordering of edges leads to permute the set of edges and analyze each of the permutations separately. Every analyzed permutation results in a numerical value. In order to generate a complexity measure, a mean value consisting of all permutation results is calculated.

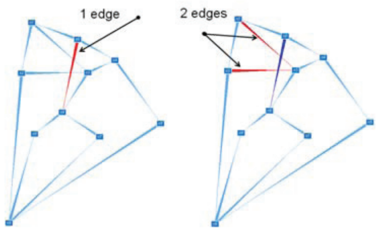

*Figure 5. The number of edges to remove depends on the order*

## 3.4 Aspects of implementation

***Permutation and computation***

To completely evaluate the mean value of all possible outcomes of the maximum planarization problem, all possible permutations from the set of edges need to be generated. The number of permutations is n! (n factorial), where n is the number of edges of the graph to be evaluated. The number of permutations grows quickly with the number of elements of the set, which renders computation almost impossible even for small graphs. Classic recursive permutation algorithms that work on standard PC hardware are not applicable already for graphs with more than 10 edges, because memory leaks will occur (i.e. the amount of information is too big to be managed by the operating system). It is possible to use multithreading and advanced iterative permuting algorithms to generate new permutations and analyze them at the same time, and this opens a possibility to compute larger structures because it is possible to deallocate memory used by finished permutations. In contrast, the recursive versions need to calculate all permutations before evaluating each of them.

***Evaluating a single permutation***

During the generation of all permutations from the set of edges, each of the created permutations can be analyzed using the edge addition method [33]. The algorithm starts with an empty set of edges. After adding the next edge from the ordering, a simple planarity-test is performed. At this point, there are two possibilities: Either the subgraph is still planar, or it is not. If the subgraph G' is planar, the algorithm adds the next ordered edge. Otherwise the subgraph G' is non-planar, the algorithm counts all the remaining edges in the ordering which are not included in the subgraph. This integer-value is stored to calculate the metric later.

***Generating a mean value***

After the algorithm is started and after e.g. 1000 permutations are available, the results of these permutations are evaluated as described above. This is done in parallel, using a so called thread-object to facilitate parallelization. After finishing the evaluation, a mean-value is calculated from the analyzed permutations and returned it to the main-method. This way, many intermediate values are calculated, from which, at the end, an overall mean that equals the resulting complexity is computed.
To reduce computation time, all thread-objects calculate mean-values for different permutations at the same time. When all permutations are done and all thread-objects finished their evaluation, the main method calculates the mean-value, using all mean-values from the thread-objects. The thread-objects enable a multiprocessor or multi-core system to run the threads at the same time, with each processor or core running a particular thread. It is even possible to run such a parallelized algorithm on a cluster of workstations. This fact is important because of the O(n!*n) complexity, i.e. of the computation time that is maximum proportional to the number of edges by the factor n!*n.

To accelerate the algorithm, it is possible to use samples (1% is an appropriate value) in order to reduce large amount of permutations. This is possible because the iterative algorithm changes the structure only very slowly, permuting one edge at a time. At 1% of the samples only one sample out of every 100 is evaluated; as the 100 different structures are all rather similar, very little accuracy is actually lost thereby, and the calculation of the mean value this way will still lead to a very good approximation of the degree of planarity, providing an average number of how many edges need to be removed to generate a planar graph.

## 4 INDEPENDENCE AND GAIN OF THE PLANARITY METRIC

### 4.1 Weyuker's criteria and the planarity metric

[21] states nine criteria for a metric to be a good metric for measuring complex systems. The paper gives quite mathematic definitions for the criteria, Therefore [35] is used for its more evidence-orientated way. Weyuker's criteria are designed to qualify software metrics as appropriate. Cardoso [35] proposed the ability of Weyuker's criteria to be used to evaluate workflow processes as well, as running a workflow is indeed similar to executing a software program. Weyuker's criteria therefore also apply to the metric here, as it is used up to now to evaluate processes as shown in figure 1. These processes are modelled *ex-post* and represent, as such, a workflow-like character.
Weyuker's criteria are the following:

1. A metric cannot measure all software programs as being equally complex.
2. There are only a finite number of programs of the same complexity.
3. Each different program may be complex.
4. The complexity of a program depends on its implementation; even if two programs solve the same problem, they can have different complexities.
5. The complexity of two programs joined together is greater than the complexity of either program considered separately.
6. If a program of a given complexity is created by joining two other programs, this does not necessarily mean that the resulting program will be of equal complexity, even if the two added program are of equal complexity.
7. A permuted version of a program can have a different complexity, i.e. the order of statements matters.
8. If a program is a straight renaming of another program, its complexity should be the same as the original program.
9. The complexity of two programs joined together may be greater than the sum of their individual complexities.

*Table 2. The proposed planarity metric and Weyuker's criteria*

| Weyuker's criteria | proposed Planarity-Metric |
|---|---|
| 1. √ | Only one single differing edge in 2 structures results in different values. |
| 2. √ | This criterion correctly used is fulfilled, but there are infinitive structures with the same complexity, if the structures differ in planar parts. It is possible to include the number of edges and the number of nodes in order to avoid this criterion. |
| 3. √ | This criterion is not affected by the evaluation concerning planarity. |
| 4. √ | The fourth criterion is fulfilled. Two different processes may do the same, but vary in their planar-metric. |
| 5. × | This criterion is not applicable considering the planar-metric. |
| 6. × | This criterion is not applicable considering the planar-metric. |
| 7. √ | This criterion addresses the basic principle of the planarity metric. |
| 8. √ | The evaluated structure is not affected by names. |
| 9. × | This criterion is not applicable considering the planar-metric. |

Additionally, a good metric should be scaled purposefully, implying that the results the metric yields should cover a significant range; for the metric proposed in this research, this holds true, as the outcome of the metric states the average number of edges that need to be removed to obtain a planar graph; as such, the results will be >1 for any non-planar graph.

Furthermore, a user needs to be aware whether the metric that is applied correlates to other available metrics. The best scenario is, of course, if all metrics are orthogonal to each other. A comparison to relevant other metrics is therefore shown in the following.

## 4.3 Case Study

In order to show the differences between the planarity metric and other structural metrics, four graphs as shown in figure 6 were analyzed. For each of them, the mean degree of cross linking, the number of cycles and the mean path length are illustrated in table 3. All graphs that were used for the examination of the proposed planarity metrics were taken from the process model shown in figure 1.

### *Setup of case study*

Each of the graphs consists of 6 nodes and 11 edges, and every couple of graph was varied in two edges. For each graph, the algorithm (implemented in Java) was run for about one hour to compute a value for the planarity based metric on a standard dual core CPU. Working with samples (only 1 percent of all permutations was taken into account) enabled the metric to calculate the complexity of a single graph with 11 edges in under one minute. The use of samples causes a variation of the results only after the second decimal place.

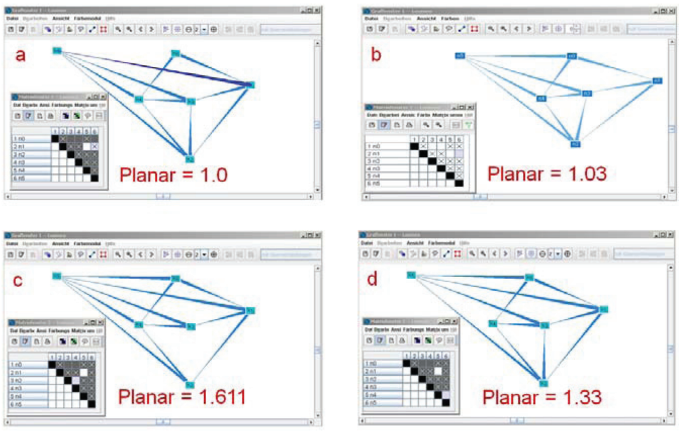

*Figure 6. Example DSMs and graphs to show differences to established structural metrics*

### *Findings and implications*

In the case study used, the complexity metric differs significantly from the other complexity metrics that were possibly correlating. At the same time, it was found that the result calculated with standard complexity metrics (number of cycles, the mean branch factor and the mean path length) remains almost identical across the four different graphs; yet, the planarity metric varies in all 4 graphs (table 3). This means, considering the described case study, the calculated informative values are different to each other. The proposed metrics lead to different complexity values. Hence, there is gain of information using the planarity metric.

*Table 3. Results of the case study*

| Graph | Planarity Metric (exact) | Planarity Metric (approximation sample size) | | | Number of Cycles | Degree of cross linking | Mean path length |
|---|---|---|---|---|---|---|---|
| | | 1% | 0.1% | 0.01% | | | |
| a | 1.0000 | 1.000 | 1.000 | 1.000 | 37 | 3.7 | 1.267 |
| b | 1.0278 | 1.023 | 1.024 | 1.016 | 39 | 3.7 | 1.267 |
| c | 1.6111 | 1.613 | 1.628 | 1.641 | 36 | 3.7 | 1.267 |
| d | 1.3287 | 1.331 | 1.350 | 1.369 | 39 | 3.7 | 1.267 |

The planarity metric was computed as an exact solution, i.e. by evaluating every single permutation as explained before. Also, it was calculated for a 1% sample, a 0.1% sample and a 0.01% sample to see the trend of the approximation. As can be clearly seen, there is a slight deviation from the exact measure. However, considering that there is a very high probability that a graph is not 100% exact, this is a very slight error. A mathematical proof of the extent, however, remains to be done.

## 5 DISCUSSION AND CONCLUSION

Structural awareness becomes more important regarding all possible domains. In different design processes it is necessary to take complexity metrics into account. Small changes in structures can cause high impacts, so all available information about structure should be used.

There are different scenarios that motivate the use of complexity metrics:

- comparing different systems at a given time to prioritize the investment of resources into e.g. rework: e.g. a process manager might be interested to compare a number of process architectures he is responsible for; to know which of them is the most complex and thus bears the highest potential to cause errors, a complexity metric is purposeful to identify the most complex process to start improvement with
- tracing changes over time to schedule possible improvements: e.g. a product architecture that is designed in a CAD system using parametric interdependencies possibly grows more and more complex during the detailing, and more and more parameters might be introduced; to trace the degree of complexity, a design engineer can employ complexity metrics to better estimate the degree of stability of his architecture
- assessing complex structures at an abstract level to estimate e.g. the amount of effort: for example in project planning, a linear timeline is desirable to guarantee smooth process execution; if, however, the tasks are interlinked in a way that no ideal sequence can be reached (as e.g. triangularization for a DSM would provide), an approximation of the degree of non-planarity of the possible process helps the planner to judge how much effort will need to be put into communication during process runtime
- identification of improvement potential, possibly serving as a fitness function in a genetic algorithm: e.g. the design of a wire harness is a most complex task, and without proper algorithmic optimization it is almost impossible to design a robust and cost-efficient architecture of the electrical wiring; to achieve an almost planar wire harness, i.e. one that has the least number of cables crossing each other, a genetic algorithm that removes single edges to see how the structure could be optimized, could use a complexity metric like the one proposed here as a fitness function; equally, if several options of removing a cable here or there, the evaluation of which structure is less complex (i.e. more planar) would prove useful
- assessment of the human cognitive ability to understand a designed system (similar to [27]): the more complex technical systems get, the more complex is also the interaction with these systems; assessing how easily a system can be comprehended (e.g. a flowchart of a process or the various states of a product and their mutual dependencies) can serve to better design a system and to judge how users will possibly interact with it; commonly, the more intertwined a system is, the harder it is to understand, therefore a planarity-based metric could well serve the purpose

There are numerous fields of application that go well beyond the examples given above. Product architectures are a common field of structural complexity management, as is process management. Equally, complex interdependencies can be found in organizational design and market structures. However, the focus of this paper was not to develop possible use cases but to extend the set of available complexity metrics in an explorative manner and prove the validity of the results using an example from process management, which could be shown even using a small example.

Future research will be twofold: On the one hand, the other gaps as identified in table 1 will be filled to complete the set of available methods to describe any given structure comprehensively.

On the other hand, the existing structural criteria and metrics will be applied to larger structures (products, processes,…) to systematically test their significance; this was not possible here for computational reasons. To this end, either a large cluster is necessary to provide an exact reference to triangulate approximations that can later serve as pragmatic descriptions for the metric, or various approximations and the trends that develop for larger structures to then deduce the average error can be used. As of now, the above described algorithm cannot handle structures exceeding 20 edges, as the biggest problem is still to generate the necessary permutations. A strategy to break down the structure into smaller computable elements is therefore under research to render the calculation more efficient.

## ACKNOWLEDGEMENTS

This work was possible as part of the Research Centre "SFB 768 - Managing cycles in innovation processes". The work was carried out in the research project A2 "Modellierung und Analyse disziplinenübergreifender Zusammenhänge", which aims at researching a methodology to assess company-wide network structures across all disciplines involved in generating innovative product-service-systems (PSS). The Research Center is funded by the German Research Foundation (DFG). The authors would like to express their gratitude for making this research possible.

## REFERENCES

[1] Deger, R. Managing Complexity in Automotive Engineering. In Lindemann, U., Danilovic, M., Deubzer, F., Maurer, M. and Kreimeyer, M., eds. *9th International DSM Conference*, p. 13-23 (Shaker, Munich, 2007).

[2] Giffin, M.L., De Weck, O.L., Buonova, G., Keller, R., Eckert, C.M. and Clarkson, P.J. Change propagation analysis in complex technical systems. *ASME 2007 International Design Engineering Technical Conferences (IDETC/CIE2007)* (ASME, Las Vegas, Nevada, 2007).

[3] Maurer, M. Structural Awareness in Complex Product Design. *Lehrstuhl für Produktentwicklung* (TU München, München, 2007).

[4] Barabási, A.-L. *Linked*. (Penguin Books, London, 2003).

[5] Boardman, J. and Sauser, B. System of Systems - the meaning of of. *System of Systems Engineering, 2006 IEEE/SMC*.

[6] Haberfellner, R., Nagel, P., Becker, M., Büchel, A. and von Massow, H., eds. *Systems Engineering: Methodik und Praxis* 2002.

[7] Cardoso, J. Approaches to Compute Workflow Complexity. *Dagstuhl Seminar „The Role of Business Processes in Service Oriented Architectures"* (Dagstuhl, Germany, 2006).

[8] Steward, D.V. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, 1981, 28, 71–74.

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

[10] Kusiak, A. *Engineering Design: Products, Processes and Systems*. (Academic Press, San Diego, 1999).

[11] Steward, D. Partitioning and Tearing Systems of Equations. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 1965, 2(2), 345-365.

[12] Danilovic, M. and Browning, T.R. Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, 2007, 25(3), 300-314.

[13] Gross, J.L. and Yellen, J. *Graph Theory and its Applications*. (Chapman & Hall/CRC, Boca Raton, 2005).

[14] Erdoes, P. and Rényi, A. On Random Graphs I. *Publicationes Mathematicae Debrecen*, 1959, 6 (290).

[15] Watts, D.J. and Strogatz, S.H. Collective Dynamics of 'Small-world' Networks. *Nature*, 1998, 393 (6684), 440-442.

[16] Barabási, A.-L. and Albert, R. Emergence of Scaling in Random Networks. *Science*, 1999, 286 (5439), 509-512.

[17] Strogatz, S.H. Exploring complex networks. *Nature*, 2001, 19 (410), 53.

[18] Albert, R., Jeong, H. and Barabasi, A.L. Error and Attack Tolerance of Complex Networks. *Nature*, 2000, 406 (6794), 378-382.

[19] Papadimitriou, C.H. *Computational Complexity*. (Addison-Wesley, Reading, MA, 1994).

[20] Ebert, C., Dumke, R., Bundschuh, M. and Schmietendorf, A. *Best Practices in Software Measurement*. (Springer, Berlin, 2005).

[21] Weyuker, E. Evaluating software complexity measures. *IEEE Transactions on Software Engineering*, 1988, 14 (9), 1357-1365.

[22] Gruhn, V. and Laue, R. Complexity Metrics for Business Process Models. *9th International Conference on Business Information Systems* (GI, Klagenfurt, Austria, 2006).

[23] Cardoso, J. Control-flow Complexity Measurement of Processes and Weyuker's Properties. *6th International Enformatika Conference*, pp. 213-218 (International Academy of Sciences,

Budapest, 2005).
[24] Summers, J. and Shah, J. Developing Measures of Complexity for Engineering Design. *DETC'03 ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (ASME, Chicago, IL, 2003).
[25] Schlick, C.M., Duckwitz, S., Gärtner, T. and Schmidt, T. A Complexity Measure for Concurrent Engineering Projects Based on the DSM. In Kreimeyer, M., Lindemann, U. and Danilovic, M., eds. *10th International DSM Conference* (Hanser, Stockholm, 2008).
[26] Kreimeyer, M., König, C. and Braun, T. Structural Metrics to Assess Processes. In Kreimeyer, M., Lindemann, U. and Danilovic, M., eds. *10th International DSM Conference*, p. 245-258 (Hansa, Stockholm, 2008).
[27] Henry, S., Kafura, D. and Harris, K. On the Relationships among three Software Metrics. *1981 ACM Workshop/Symposium on Measurement and Evaluation of Software Quality*, p. 81-88 University of Maryland, 1981.
[28] Shao, J. and Wang, Y. A New Measure of Software Complexity based on Cognitive Weights. (*IEEE Canadian Conference On Electrical And Computer Engineering*, p. 1333-1338 2003).
[29] Kuratowski, C. *Sur le problème des courbes gauches en topologie*. 1930.
[30] John, H. and Robert, T. Efficient Planarity Testing. *J. ACM*, 1974, 21 (4), 549-568.
[31] Booth, K.S. and Lueker, G.S. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithm. *Journal of Computational Systems Science*, 1976, 335-379.
[32] Shih, W.K. and Hsu, W.L. A new planarity test. *Theoretical Computer Science,* 1999, 223.
[33] Boyer, J.M. and Myrvold, W.J. On the Cutting Edge: Simplified O(n) Planarity by Edge Addition. *Journal of Graph Algorithms and Applications*, 2004, 8, 241–273.
[34] Chiba, T., Nishioka, I. and Shirakawa, I. An Algorithm of Maximal Planarization of Graphs. *IEEE Syrup. on Circuits and Systems*, p. 649-652, 1979).
[35] Cardoso, J. How to Measure the Control-flow Complexity of Web Processes and Workflows. In Fischer, L., ed. *The Workflow Handbook,* pp. 199-212 (Future Strategies Inc., Lighthouse Point, FL, 2005).

Contact: S. Kortler
Institute of Product Development, Technische Universität München
Boltzmannstr. 15
85748 Garching
Germany
Phone +49.89.28915153
Fax +49.89.28915144
E-mail sebastian.kortler@pe.mw.tum.de

Sebastian Kortler is a scientific assistant at the Institute of Product Development at the Technische Universität München, Germany since 2008. Before, he studied computer science until 2007. Currently, his research is focused on structural complexity and the development of structural characteristics that govern complex systems.

Matthias Kreimeyer is a scientific assistant at the Institute of Product Development at the Technische Universität München, Germany. His research is focused on structural complexity and the management of engineering design processes; as part of his work, he is co-chair of the Special Interest Group "Managing Structural Complexity" and has organized the DSM Conference since 2007.

Udo Lindemann is a full professor at the Technische Universität München, Germany, and has been the head of the Institute of Product Development since 1995, having published several books and papers on engineering design. He is committed in multiple institutions, among others as president of the Design Society and as an active member of the German Academy of Science and Engineering.