Towards the Characterization of Realistic Models: Evaluation of Multidisciplinary Graph Metrics[*]

Gábor Szárnyas
szarnyas@mit.bme.hu

Zsolt Kővári
zsolt.kovari@inf.mit.bme.hu

Ágnes Salánki
salanki@mit.bme.hu

Dániel Varró
varro@mit.bme.hu

Budapest University of Technology and Economics
Department of Measurement and Information Systems
MTA-BME Lendület Research Group on Cyber-Physical Systems

ABSTRACT

Custom generators of graph-based models are used in MDE for many purposes such as functional testing and performance benchmarking of modeling environments to ensure the correctness and scalability of tools. However, while existing generators may generate large models in increasing size, these models are claimed to be simple and synthetic, which hinders their credibility for industrial and research benchmarking purposes. But how to characterize a realistic model used in software and systems engineering? This question is investigated in the paper by collecting over 17 different widely used graph metrics taken from other disciplines (e.g. network theory) and evaluating them on 83 instance models originating from six modeling domains. Our preliminary results show that certain metrics are similar within a domain, but differ greatly between domains, which makes them suitable input for future instance model generators to derive more realistic models.

1. INTRODUCTION

Context. While empirical software engineering highly relies on the source code repositories of large open-source projects, scalability assessment of model-driven engineering (MDE) tools on large models has been much more problematic. On the one hand, real complex industrial models are rarely available to public as intellectual property rights of all parties need to be protected. On the other hand, faithfulness of scalability evaluations using synthetic, auto-generated models are frequently considered as questionable. Anyhow, there is an increasing interest in model generators to be used for validating, testing or benchmarking MDE tools with advanced support for queries and transformations [18, 10, 2].

[*]©ACM, 2016. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in MODELS 2016, http://models2016.irisa.fr/

MODELS '16, October 02 - 07, 2016, Saint-Malo, France

ACM ISBN 978-1-4503-4321-3/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2976767.2976786

Objective. As a *long-term objective* and major research challenge, we aim at generating domain-specific graph models that are scalable, realistic, consistent and diverse *at the same time.*

Problem. *But what makes a model realistic?* Any engineer can distinguish an auto-generated model from a manually designed model by inspecting attributes (e.g. names). But what if we abstract from all attributes of the model and inspect only the (typed) graph structure? How can we characterize and distinguish systems engineering models (e.g. Capella [32], AutoFOCUS [3]) from models reverse engineered from source code, for instance?

Method. We identify and evaluate graph-based model metrics known from other disciplines to decide which can best describe the characteristics of real models taken from software and systems engineering. We calculate these metrics on 83 models, and carry out an initial evaluation using statistical and visual exploratory data analysis techniques. The output of our evaluation includes recommendations on characteristic metrics and potential hints for constructing future model generators of realistic domain-specific models.

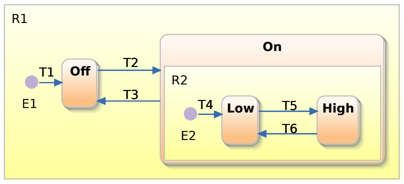
We reuse several graph metrics of network theory [30, 38, 15] already used in other disciplines (e.g. physics, biology, social network analysis) to reveal hidden structural properties of complex systems, and observe structural differences between them. However, since most of the analyzed networks are one-dimensional (untyped), i.e. they only contain edges of a single dimension (type), their direct adaptation to MDE models may not be sufficient due to their strongly typed nature. Therefore, we also collected and evaluated several graph metrics for multidimensional (multi-typed) networks [31]. Such *multidimensional metrics* [6, 4, 31] express structural properties with respect to a dimension, and how different dimensions emerge together.

Contributions of the paper. This paper presents the following specific contributions:

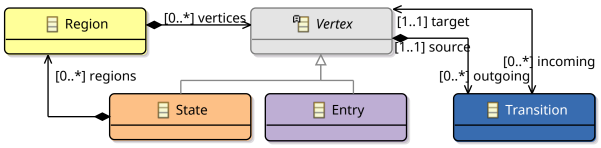
- We collected 17 graph metrics from different disciplines (mostly network theory).
- We evaluated these metrics over 6 different modeling tools (domains) on 83 real models.
- We carried out statistical and visual exploratory data analysis to identify characteristic metrics.

2. PRELIMINARIES

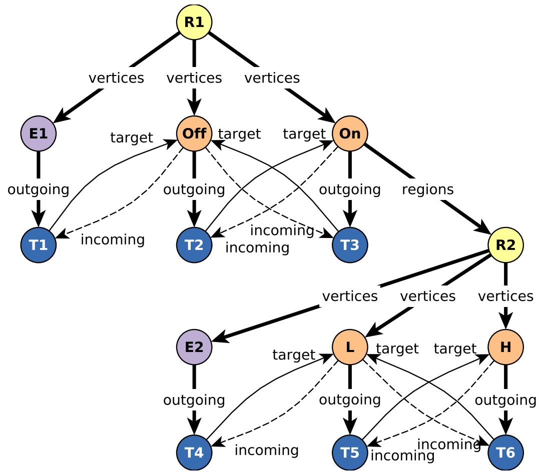
Our running example is a statechart (Figure 1a) describing the behaviour of a light switch. The statechart is defined over a simplified metamodel of Yakindu [40] (Figure 1b).



(a) Light switch statechart.



(b) Simplified statechart metamodel.



(c) Multidimensional graph of the statechart. Containment edges are marked with thicker lines.

Figure 1: The statechart example, the statechart metamodel and the multidimensional graph representing the statechart.

To calculate the metrics of models, we map them to multidimensional graphs [5]. A multidimensional graph is defined as $G = (V, E, D)$, where V is the set of nodes (vertices), E is the set of directed edges between the nodes, and D represents the set of *dimensions* that label the edges. An edge is defined with a triple $(v, w, d) \in E$, where $v, w \in V$ and $d \in D$ is the dimension of the edge.

Mapping. Each model is an instance of a metamodel defined in Ecore, the metamodeling language of EMF (Eclipse Modeling Framework) [35]. The models are mapped to multidimensional graphs using the following rules:

- Each object is mapped to a node ($v \in V$).
- Each reference type is mapped to a dimension ($d \in D$).
- Each reference instance is mapped to a directed edge between nodes, $(v, w, d) \in E$. Nodes v and w are the corresponding nodes of the reference's source and target objects. The dimension d is determined by the reference type.
- Object types are omitted. As a consequence, the graph does not contain information on the classes in the model.
- Derived references and attributes are omitted from the graph. Opposite edges of containment references are also removed (see Section 4.2 for details).

Figure 1c shows the statechart model as a multidimensional graph. The dimensions include the reference types vertices, outgoing, incoming, target and regions (while excluding the source dimension as it is the inverse of the outgoing containment reference.) The objects and references in the instance model are mapped to nodes and edges of the graph.

Basic concepts in multidimensional graphs. Nodes $v, w \in V$ are *connected* in a dimension $d \in D$ if they have an edge in that dimension. Formally,

$$Connected(v, w, d) \iff (v, w, d) \in E \lor (w, v, d) \in E.$$

A node $v \in V$ is *active* in a dimension $d \in D$ if the node has at least one connection in that dimension:

$$Active(v, d) \iff \exists w \in W : Connected(v, w, d)$$

In the statechart graph (Figure 1c), node E1 is *connected* to node R1 along vertices and to T1 along outgoing. Hence, node R1 is *active* in dimensions vertices and outgoing.

Note on terminology. Different terminologies exist in the literature for multidimensional networks [6], such as multi-layered networks [8, 9] and multiplex networks [31, 4], as summarized in [23].

3. GRAPH METRICS

We collected single one-dimensional metrics [15], and multidimensional metrics successfully used in other fields of science to characterize multidimensional graphs. Our assumption is that they can likely be useful in the context of software and systems models. We also demonstrate the metrics on our example model (Figure 1c).

3.1 One-Dimensional Metrics

One-dimensional metrics include the number of nodes $n = |V|$ and the number of edges $e = |E|$. For a node $v \in V$, the *in-degree* k_v^{in} is the number of incoming edges and the *out-degree* k_v^{out} is the number of outgoing edges. The degree k_v equals to the total number of the incoming and outgoing edges of node v, i.e. $k_v = k_v^{in} + k_v^{out}$. The average degree of a graph is $\langle k \rangle = \frac{2e}{n}$, where $\langle\rangle$ denotes the average.

The **clustering coefficient** $C(v)$ measures the probability that the neighbours of a node $v \in V$ are also connected to each other [38]. Formally, it is calculated as

$$C(v) = \frac{2e_v}{k_v(k_v - 1)},$$

where e_v denotes the number of connected pairs among the neighbours of v. $C(v)$ is normalized to the interval $[0, 1]$, equalling to 1 if every neighbour of v is connected to each other and to 0 if there are no connections between the neighbours of v. In short, $C(v)$ measures the ratio of existing/possible triangles (subgraphs with 3 nodes and 3 edges) containing node v.

In the example model (Figure 1c), $|V| = 14$, $|E| = 25$, $|D| = 5$, $k_{R2}^{in} = 1$, $k_{R2}^{out} = 3$ and $\langle k \rangle = 3.57$. The clustering coefficient is 0 for every node in the graph due to the absence of triangles.

Name	**Notation**	**Scope**	**Nd.**
Dimensional degree [6]	$\textit{Degree}\,(v,d)$	D/N	○
Node dimension activity [31]	$\textit{NDA}(d)$	D	○
Node dimension connectivity [6]	$\textit{NDC}(d)$	D	●
Node exclusive dimension conn. [6]	$\textit{NEDC}(d)$	D	●
Edge dimension activity	$\textit{EDA}(d)$	D	○
Edge dimension connectivity [6]	$\textit{EDC}(d)$	D	●
Node activity [31]	$\textit{NA}(v)$	N	○
Multiplex participation coeff. [4]	$\textit{MPC}(v)$	N	●
Dimensional clustering coeff. [4]	$\textit{DC}(v)$	N	●
Pairwise multiplexity [31]	$Q(d_1,d_2)$	D^2	●

Table 1: Summary of metrics for multidimensional graphs (Scope: N = Nodes, D = Dimensions, D^2 = Dimension pairs; Nd: Normalized; $v \in V$; $d, d_1, d_2 \in D$)

3.2 Multidimensional Metrics

The *Scope* column lists whether the metric is interpreted on *Dimensions, Nodes,* both or *Dimension pairs.* We also denote if a metric is normalized, i.e. it takes values in $[0, 1]$.

3.2.1 Metrics for Dimensions and Nodes

For a node $v \in V$ and a dimension $d \in D$, the **dimensional degree** [6] is the number of neighbours of v with respect to dimension d. Formally,

$$\textit{Degree}\,(v,d) = \left|\left\{w \in V \,\middle|\, \textit{Connected}(v,w,d)\right\}\right|.$$

The metric is also defined for a set of a dimensions $\widehat{D} \subseteq D$:

$$\textit{Degree}\left(v,\widehat{D}\right) = \textstyle\sum_{d\in\widehat{D}} \textit{Degree}\,(v,d).$$

If $\widehat{D} = D$, the dimensional degree of a node v equals to its one-dimensional degree k_v as every dimension is aggregated.

In the example graph (Figure 1c), $\textit{Degree}\,(\mathsf{T3},\mathsf{target}) = 1$, and $\textit{Degree}\,(\mathsf{R2},\mathsf{vertices}) = 3$.

3.2.2 Metrics for Dimensions

Node dimension activity ($\textit{NDA}$) (introduced as *layer activity* in [31]) characterizes a dimension $d \in D$, and equals to the number of nodes that are active in dimension d:

$$\textit{NDA}(d) = \left|\{v \in V \mid \textit{Active}(v,d)\}\right|.$$

Node dimension connectivity ($\textit{NDC}$) [6] computes the ratio of nodes in the network that belong to dimension d:

$$\textit{NDC}(d) = \frac{\textit{NDA}(d)}{|V|}.$$

Node exclusive dimension connectivity ($\textit{NEDC}$) [6] is similar to *node dimension connectivity*, but it calculates the ratio of nodes that belong *exclusively* to dimension d. In other words, it calculates the ratio of nodes that do not have other dimensions than d.

Edge dimension activity ($\textit{EDA}$) determines the number of edges that belong to a dimension $d \in D$:

$$\textit{EDA}(d) = \left|\{(v,w,d) \in E \mid v,w \in V\}\right|.$$

Edge dimension connectivity ($\textit{EDC}$) [6] determines the ratio of edges labeled with dimension $d \in D$:

$$\textit{EDC}(d) = \frac{\textit{EDA}(d)}{|E|}.$$

In the example graph (Figure 1c):

- $\textit{NDA}(\mathsf{outgoing}) = 12$ and $\textit{NDC}(\mathsf{outgoing}) = 0.85$ implying that the majority of nodes are active in dimension $\mathsf{outgoing}$. However, there are no nodes exclusively active in this dimension, hence $\textit{NEDC}(\mathsf{outgoing}) = 0$.
- $\textit{EDA}(\mathsf{outgoing}) = 6$ and $\textit{EDC}(\mathsf{outgoing}) = 0.24$, meaning that 24% of edges are in dimension $\mathsf{outgoing}$.

3.2.3 Metrics for Nodes

Node activity ($\textit{NA}$) [31] identifies the number of dimensions in which node $v \in V$ is active. Formally,

$$\textit{NA}(v) = \left|\{d \in D \mid \textit{Active}(v,d)\}\right|.$$

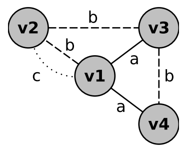
The **multiplex participation coefficient** ($\textit{MPC}$) [4] measures whether the connections of node $v \in V$ are uniformly distributed among dimensions D:

$$\textit{MPC}(v) = \frac{|D|}{|D|-1}\left[1 - \sum_{d\in D}\left(\frac{\textit{Degree}(v,d)}{\textit{Degree}(v,D)}\right)^2\right].$$

$\textit{MPC}(v)$ takes values in $[0, 1]$, equalling to 0 if all the edges of v belong to a single dimension, and to 1 if v has exactly the same number of edges on each of dimensions D.

In the example graph (Figure 1c), $\textit{NA}(\mathsf{L}) = 4$, as node L is active in 4 dimensions. $\textit{MPC}(\mathsf{R1}) = 0$ meaning that all the edges belonging to $\mathsf{R1}$ are from a single dimension ($\mathsf{vertices}$), while $\textit{MPC}(\mathsf{R2}) = 0.46$ implying that more dimensions belong to node $\mathsf{R2}$ and they are not uniformly distributed.

Dimensional clustering coefficient ($\textit{DC}$) measures the ratio of multidimensional triangles centered in a node $v \in V$. We use three definitions of $\textit{DC}$, denoted by $\textit{DC}_1$, $\textit{DC}_2$, and $\textit{DC}_3$. Each one is normalized to $[0, 1]$ by the possible number of triangles centered in v. To demonstrate the definitions, we use the example graph in Figure 2.



	v_1	v_2	v_3	v_4
$C(v)$	0.67	1.00	0.67	1.00
$DC_1(v)$	1.00	0.50	0.00	0.00
$DC_2(v)$	0.25	1.00	0.50	0.00
$DC_3(v)$	0.17	0.33	0.50	0.33

Figure 2: Example for the clustering coefficient metrics.

$DC_1(v)$ [4] considers triangles where dimensions between node v and its neighbours are the same, but the dimension between the neighbours is different. $DC_1(v_1) = 1$ as only v_3 and v_4 are connected to v_1 on the same dimension (a), while v_3 and v_4 are connected on a different dimension (b).

$DC_2(v)$ [4] considers triangles with 3 different dimensions. The edges of v_1 could be completed to a three-dimensional triangle 4 ways:

1. (v_1, v_2, b), (v_1, v_3, a) with (v_2, v_3, c)
2. (v_1, v_2, c), (v_1, v_3, a) with (v_2, v_3, b) (this one exists)
3. (v_1, v_2, b), (v_1, v_4, a) with (v_2, v_4, c)
4. (v_1, v_2, c), (v_1, v_4, a) with (v_2, v_4, b)

From these, only one triangle exists in the graph ($v_1v_2v_3$ on dimensions c, a, b), hence $DC_2(v_1) = \frac{1}{4} = 0.25$.

We also introduce $DC_3(v)$, a slight variation of DC_2. DC_3 considers triangles where the dimension between the neighbours is not necessarily different from the dimension between v and its neighbours. For example, $DC_3(v_1) = \frac{2}{12} = 0.17$, since there are 12 possible triangles around v, but only two exist in the graph ($v_2v_1v_3$ on dimensions b, a, b and c, a, b).

3.2.4 Metrics for Dimension Pairs

Pairwise multiplexity (Q) [31] is defined for a pair of dimensions, $d_1, d_2 \in D$. Its value determines the ratio of nodes from the network, which are active in both dimensions d_1 and d_2. Intuitively, the more mutual nodes the two dimensions have, the higher their pairwise multiplexity is.

The *node activity* binary vector a_v ($v \in V$) is defined as:

$$a_v = \left\{ a_v^{[1]}, a_v^{[2]}, \dots, a_v^{[|D|]} \right\}, \text{where } a_v^{[d]} = \begin{cases} 1, & \text{if } Active(v,d), \\ 0, & \text{otherwise.} \end{cases}$$

Using this vector, the *pairwise multiplexity* metric is

$$Q(d_1, d_2) = \frac{1}{|V|} \sum_{v \in V} a_v^{[d_1]} a_v^{[d_2]}.$$

$Q(d_1, d_2)$ takes values from the $[0, 1]$ interval, and equals to 1 if the activity vectors $a_v^{[d_1]}$ and $a_v^{[d_2]}$ are identical when d_1 and d_2 belong to the same nodes.

In the example graph (Figure 1c), $Q(\mathsf{incoming}, \mathsf{outgoing}) = 0.71$, as these two dimensions often appear together. This can be explained by the fact that every State node belongs to both dimensions. However, the value is less than 1 as Entry nodes are never active in dimension incoming.

4. EXPERIMENTAL SETUP

We analyzed the characteristics of 83 graph models by evaluating single and multidimensional metrics on them.

4.1 Domains and Instance Models

Models were taken from six different domains:

- *AutoFOCUS* [3] is an MDE systems engineering tool for designing distributed, embedded software systems.
- *Building Information Model (BIM)* [16] is a representation format for architecture designs. BIM models were provided by Uninova, an industrial partner in the MONDO EU FP7 project [28].
- *Capella* [32] is a graphical modeling workbench for model-based systems engineering developed at Thales to support the Arcadia engineering method.
- *JaMoPP* [20] parses Java source code into EMF-based models and vice versa by constructing abstract syntax trees (ASTs) from the source code with the extension of cross-references (e.g. method calls, variable access).
- *Yakindu Statecharts Tools* [40] is an integrated modeling environment developed by Itemis AG. It can be used for the specification and development of reactive, event-driven systems using statecharts.
- The *Train Benchmark* [36] is a benchmark with contributions from several authors of this paper to measure the performance of continuous model validation on graph-based models in a railway system domain that originates from the MOGENTES EU FP7 [27] project. We used 4 synthetic models from the Train Benchmark in experiments, while all models from other domains were real models created by engineers.

Table 2 shows the basic graph characteristics of the models. Each domain contains several instance models (3–34) with different sizes, where BIM and JaMoPP models are the largest (up to 10M nodes). The average degree $\langle k \rangle$ ranges from 2.2 to 7.8 which shows a significant difference between our models and social networks [8, 14] where the average degree is often ten times as much. The number of dimensions $|D|$ varies across domains: while Yakindu or Train Benchmark models are built from 4–12 dimensions, Capella models of similar size may contain 10 times more dimensions. The ratio of containment edges, EDC $(ctm.)$, is higher for AutoFOCUS and JaMoPP models which means fewer cross-references between the objects. This also explains the smaller average degrees for these models. Note that the BIM models are flat graphs without a containment hierarchy.

Domain	#	$\lvert D\rvert$	$\lvert V\rvert$	$\langle k \rangle$	EDC $(ctm.)$
AutoF.	24	16 – 74	10 – 1k	2.2 – 3.2	0.7 – 0.92
BIM	34	51 – 117	10k – 10M	2.2 – 5.2	0 – 0
Capella	3	103 – 182	1k – 10k	4.2 – 5.0	0.41 – 0.48
JaMoPP	9	67 – 98	100k – 1M	2.6 – 2.6	0.8 – 0.8
Yakindu	9	4 – 4	10 – 1k	3.2 – 4.6	0.41 – 0.52
Train B.	4	12 – 12	1k – 10k	7.2 – 7.8	0.16 – 0.16

Table 2: Characteristics of the instance models. #: number of instance models; $|D|$: number of dimensions; $|V|$: number of nodes; $\langle k \rangle$: average degree; $EDC(ctm.)$: ratio of containment edges.

4.2 Data Preparation

When we first evaluated the multidimensional metrics on these models, we observed outlying values along several distribution functions, which were caused by some extremities of models. Therefore, we carried out some data preparation and cleansing prior to the actual data analysis below.

Omitting layout information. Some modeling tools (AutoFOCUS and Yakindu) persisted graphical information of diagram elements to the model itself. As several metrics were dominated by the large number of such elements, we decided to remove the layout information from these models (except for BIM where graphics is the key information in the models).

Omitting models of extreme sizes. We omitted models that were very small (e.g. overly simple example models) compared to all other models of the domain and therefore distorted metric values and thus the results of the analysis.

No derived edges. All derived edges were removed, including inverse edges of containment (see Section 2).

5. EVALUATION

We calculated the values of each multidimensional metric of Section 3 for every instance model which yielded over 160 million data records as input for our analysis. This paper only contains plots related to some interesting findings while data sets and detailed plots are available online.[1]

Below, we investigate three research questions, which are highly important for (i) understanding the structural differences between real vs. synthetic models and (ii) parameterizing future model generators to create realistic models.

5.1 Which Metrics Are Characteristic?

For describing the structure of models, we consider a metric *characteristic* if it has both of the following properties.

[1]See http://docs.inf.mit.bme.hu/model-metrics/.

- **Homogeneity:** models *within* a specific domain have similar distribution in this metric.
- **Distinctiveness:** models from *different domains* can be distinguished based on their distribution in this metric.

Metrics ranking high in one aspect do not necessarily perform well in the other one: values belonging to even very narrow ranges can overlap entirely with each other (indicating indistinguishable domains) and diverse domains can be separated efficiently, if they are different enough.

Table 3 contains the *homogeneity values* for each metric–domain pairs. Cells with black background indicate that the metric is highly homogeneous within a certain domain, while white cells mark that it is heterogeneous. Grey cells usually indicate domains containing outlier models which do not fall into previous categories (with homogeneity values between 0.3 and 0.7). For example, AutoFOCUS and Yakindu models are heterogeneous along each dimension-related metric, while Train Benchmark models are highly homogeneous here (as expected) due to their synthetic nature.

Name	**AutoF.**	**BIM**	**Capella**	**JaMoPP**	**Train B.**	**Yakindu**
Dimensional c. c. 1	0.06	0.00	0.04	0.00	0.02	0.00
Dimensional c. c. 2	0.21	0.01	0.27	0.14	0.02	0.00
Dimensional c. c. 3	0.21	0.01	0.27	0.14	0.02	0.00
Clustering c.	0.21	0.28	0.19	0.14	0.02	0.00
Multiplex p. coeff.	0.55	0.78	0.30	0.42	0.01	0.29
Pairwise multiplexity	0.60	0.39	0.30	0.21	0.41	0.52
Node dimension conn.	0.86	0.63	0.42	0.33	0.42	0.50
Node exclusive dim. conn.	0.86	0.63	0.42	0.33	0.42	0.50
Node dim. activity	0.86	0.63	0.42	0.33	0.42	0.50
Edge dim. connectivity	0.82	0.59	0.40	0.33	0.42	0.76
Node activity	0.98	0.96	0.64	0.51	0.01	0.12
Degree list	0.99	0.99	0.92	0.99	0.99	0.99

Table 3: Summary of metric homogeneity (see Section 5.4).

Figure 3 summarizes the *distinctiveness* of metrics for each pair of domains. Red cells indicate that a certain domain pair can be separated with a high confidence using the metric (e.g. by visually inspecting the shape of their distribution or applying unsupervised learning algorithms), while black cells indicate indistinguishable domain pairs. For example, Capella and JaMoPP models have similar characteristics in edge-related metrics such as *NDC* and *Q* but can be distinguished based on their *MPC* distribution.

In Table 3 and Figure 3, the metrics are ranked by homogeneity and distinctiveness. DC_2 and DC_3 rank high in both properties, which makes them the *best candidates for domain characterization*. Models of real domains are entirely homogeneous in DC_1 due to the dominance of zero values (99-100%) in their distributions. Therefore, it is not useful for distinguishing domains in general, even if some models have shown completely different values (e.g. Train Benchmark models because of their tightly connected structure). Some metrics, e.g. k, perform poorly in both properties.

Figure 4 presents the distribution functions of two extrema DC_2 (left) and k (right). Except for AutoFOCUS models, the domains are mainly distinguishable using DC_2. Inability of k for characterization is clearly noticeable in the figure: the distributions between different domains overlap significantly, making separation impossible.

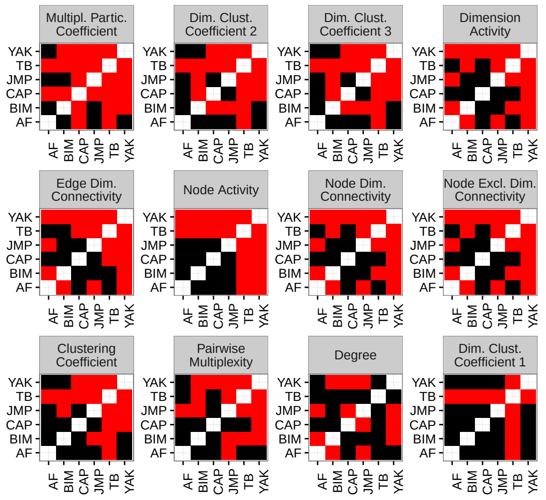


Figure 3: Summary of metric distinctiveness (Section 5.4). AF: AutoFOCUS, CAP: Capella, JMP: JaMoPP, TB: Train Benchmark, YAK: Yakindu.

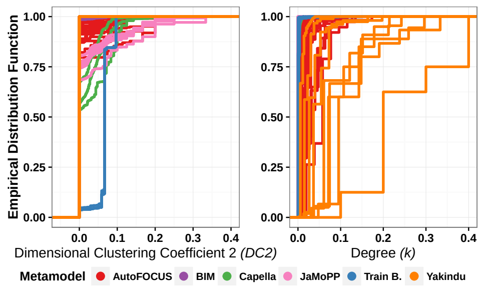


Figure 4: Distributions of DC_2 and k values.

5.2 How Do Domains Differ?

Some metrics turned out to be useful also for describing models by revealing structural characteristics or hidden properties. During the analysis, we made the following domain-specific observations.

Clusteredness. The metrics indicating the number of triangles have significant differences across domains. Yakindu and Train Benchmark models represent the two extrema: while the former ones have almost exclusively zero C values resulting in an average value of 0.008, the latter ones have an average of 0.38. This is caused by the structural properties of the Train Benchmark [36]: railway segments and their sensors are tightly connected leading to many triangles.

Dominant dimensions. All domains contain a small set of dominant dimensions, with at most four of them covering 80% of the graph. In particular, BIM and JaMoPP models contain a single dimension covering 40-50%. For BIM, these edges encode the layout of the buildings, while for JaMoPP, they form the containment hierarchy following the AST.

Dominance of containment edges. We categorized dimensions by splitting them in two groups indicating whether they represent a containment relation or not. *Containment edges* are the structural building blocks of models in software and systems engineering, while *non-containment edges* represent other semantic information between model elements.

We found that the ratio of containment edges vary drastically across the domains, e.g. it is approx. 45% in case of Capella and 80% in case of JaMoPP models. Moreover, there is no obvious relationship between the ratio of containment types in the metamodel and the containment edges in the instance models, thus metamodels in themselves are insufficient sources for characterizing realistic models.

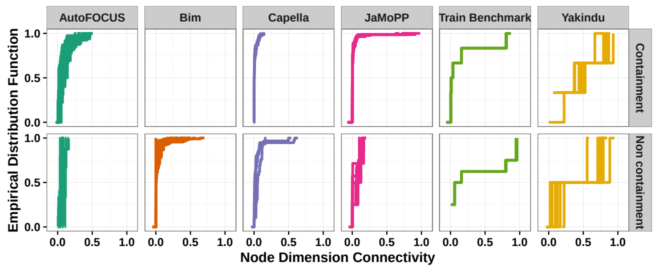


Figure 5: Distribution of NDC values in containment and non-containment subnetworks.

Figure 5 shows the NDC values for the containment and non-containment dimensions. For real models, there are obvious differences in the non-containment subnetworks, only the synthetic Train Benchmark models provide identical characteristics. We observed significant differences even in the containment dimensions for AutoFOCUS and Yakindu models. Although BIM models have no explicit containment edges, some of their dimensions provide extreme similarity to containment subnetworks in other models, e.g. JaMoPP.

5.3 What Makes a Model Realistic?

As one of our long-term research objectives is to generate realistic instance models, we attempted to identify a set of metrics which are able to capture the characteristics of real models. To achieve this, we compared real and synthetic instance models from four domains (AutoFOCUS, Capella, JaMoPP and Yakindu). We created synthetic models with a random model generator, using the following approach:

1. We removed all edges from the model that are not part of the containment hierarchy.
2. For each removed edge, we inserted a new edge of the same type. The start and end nodes of the new edge were chosen using a pseudorandom generator with a uniform distribution from the nodes which fulfill the type constraints prescribed by the edge type.

Compared to a fully random model generator, our setup presents a more adverse situation for a metrics-based distinction since a significant part of the models remains real.

We calculated the metrics on both real and synthetic models. Figure 6 illustrates the influence of randomization on two metrics, C and Q in Capella models. We found that while the majority of metrics did not show any particular difference (see e.g. Q in the right part of Figure 6), clustering metrics such as C, DC_2 and DC_3 showed significant changes both in their ranges (the maximum decreased) and distribution. This change may be explained by the fact that randomization decreased the clusteredness of the graph, as the randomly inserted edges are less likely to form a triangle (compared to a model designed by a domain expert). We observed this phenomenon in each domain with a strength depending on the number of removed edges. Thus, it was less drastic in Yakindu instances than in large models.

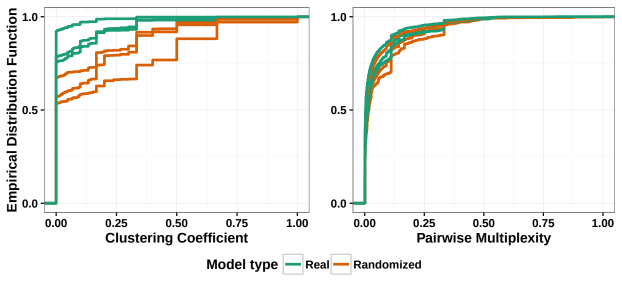


Figure 6: The distribution of C values (left) is significantly different between the real and randomized version of a particular model.

5.4 Statistical Methodology

In order to objectively characterize homogeneity and distinctiveness, we needed a concept describing the similarity of model pairs for a certain metric. We compared the whole distributions of values (and not only their descriptive summary like mean or variance) and we chose Kolmogorov-Smirnov statistic (KS) [24] as a distance measure of models.

The KS statistics quantifies the maximal difference between the empirical distribution function lines at a given value. It is sensitive to both shape and location differences, it takes a 0 value only if the distributions are identical, while it is 1 if the values of models are in disjunct ranges (even if their shapes are identical).

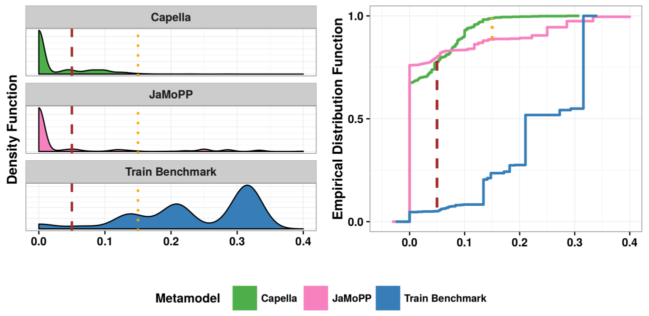


Figure 7: Comparing model pairs with KS distance in DC_2.

Figure 7 illustrates the KS distance of DC_2 distributions of models originating from three different domains (density functions on the left, empirical distribution functions on the right). KS distance between JaMoPP and Train Benchmark is 0.8, reflecting that their lines are far from each other: while 74% of JaMoPP values are smaller than 0.05 (this is the value where the difference is the largest, marked with the red dashed line), 90% of Train Benchmark values lay above this threshold. On the contrary, the maximal distance between Capella and JaMoPP models is only 0.1 (orange dotted line). Based on this distance function, we defined homogeneity and distinctiveness as the following.

Homogeneity of a domain is calculated as the ratio of the maximal KS distance within the domain and the maximal distance across each model pair; it is 0 if every model of the domain has an identical distribution and 1 if this domain spans across the entire metric space.

Distinctiveness of domain pairs is calculated as the average *membership confidence* of their models, which is the ratio of domain-identical instances (i.e. models from the same domain) in its k^{th} neighbourhood, using the idea of *kNN* classification methods [39]. Distinctiveness is 1 if the minimal inter-cluster distance is larger than each k^{th} intra-cluster distance and decreases with every pair of models, which, while belonging to different domains, produce a smaller distance to each other than to their domain-identical neighbours.

Table 3 contains the homogeneity values of the domains. Distinctiveness is computed with a k of 2, cells of Figure 3 are colored red if their distinctiveness is 1.

5.5 Threats to Validity

Metrics cannot capture semantics. The metrics used in this paper describe the structure of the models, thus they cannot explicitly express semantic content. However, the semantics of different domains may be captured in a significantly different way, which further hinders the characterization from different domains.

How real are our models? We used a variety of sources to gather models for analysis. BIM models are real models obtained from an industrial partner. JaMoPP models are generated from open-source code repositories, this way they are real large models. For Capella, AutoFOCUS and Yakindu, we used openly available tutorial models provided by the tool developers themselves who are experts in their domain. As an intentional exception, the Train Benchmark models are fully synthetic as they were created by the model generator of the benchmark.

5.6 Summary of Findings

Our analysis provides some insights that need to be considered in future generators to synthesize realistic models.

1. Relying only upon metamodel-level information is clearly insufficient, real instance models of human engineers are required to characterize the domain.
2. Containment edges frequently dominate distributions, which necessitates data preparation (Section 4.2). However, such edges can be exploited for model generation.
3. Many edges follow the locality principle, i.e. they often lead to neighbouring nodes (and not to distant ones).
4. Characteristic metrics can be used as an objective function for a search-based model generator.

6. RELATED WORK

Reuse of metrics. A collection of multidimensional metrics is defined in [6, 31, 4] where the authors study the expressiveness of their metrics on real-life networks from heterogeneous domains e.g. from a social network (Flickr), co-authorship data (DBLP), query log analysis, social, engineering and biological networks. In this paper, we reuse the metrics proposed in [6, 31, 4] and evaluate them on models taken dominantly from software and systems engineering.

Comparative studies of networks from other disciplines. Revealing essential structural similarities and differentiations among networks from different domains is a fundamental objective in network theory. Such studies [1, 14] characterize a diverse set of models derived from different domains. However, these studies are carried out on one-dimensional networks. So far, existing multidimensional studies only focused on a single application domain, such as neighbourhood and centrality analysis of a Polish social network [8], relevance and correlation analysis of different dimensions in Flickr [22], community detection in the network of YouTube [37], analysis of co-authorship in the DBLP network [9] and characteristics of different transportation networks (European Air Network [12, 11], cargo ship movements [21]). In this paper, we aim to find structural differences between MDE models of different domains by taking *multidimensionality* into account.

Use of network analysis in software engineering. The authors of [7] use graph metrics to capture the structure and evolution of software products and processes in order to detect significant structural changes, help estimate bug severity, prioritize debugging efforts, and predict defect-prone releases in software engineering. Additionally, the principles of complex networks are used to measure the structural complexity of software systems [25, 26] and to predict defects on dependency graphs [41]. Our motivation is to find metrics that are able to characterize and distinguish models used in tools of software and systems engineering.

Metrics in MDE. Our research group investigated the correlation between model query performance and metrics describing the queries and the models [19]. The authors of [33] use metrics to understand the main characteristics of domain-specific metamodels and to study model transformations with respect to the corresponding metamodels, and search correlations between them via analytical measures [34]. A generic σ-metric is proposed in [29] to assist in empirically validating various quality attributes. Finally, several approaches exist to define metrics using high-level constraint languages [13, 17]. The main novel aspect of our work is to identify characteristic graph metrics for describing *real instance models* on a statistical basis to help develop future model generators.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we identified several graph metrics known from other disciplines and evaluated them on 83 instance models of 6 different tools dominantly from software and systems engineering domains in order to identify *characteristic* metrics using statistical and visual data analysis techniques. We consider a metric characteristic if it separates models of different domains from each other, while provides similar values for models within the same domain. We also discussed whether some of these metrics can *distinguish real models from auto-generated synthetic ones*, which is the first investigation of graph models for such a purpose up to our best knowledge. Our initial finding is that different versions of clustering coefficients were particularly useful for such classifications. But, unsurprisingly, no single metric was able to sufficiently handle all the domains.

In the future, we plan to use our findings for (i) developing domain-specific model generators that are capable of synthesizing realistic models, and (ii) fine-tuning search-plan based graph query optimization techniques. We also need to improve the performance of computing some metrics, which turned out to be time-consuming when computing metrics which take different values for each node-dimension pair.

Acknowledgments

This work was partially supported by the MONDO (EU ICT-611125) project and the MTA-BME Lendület Research Group on Cyber-Physical Systems. We would like to thank István Ráth for his ideas and suggestions on the topic.

8. REFERENCES

[1] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74(1):47–97, 2002.

[2] V. Aranega et al. Towards an automation of the mutation analysis dedicated to model transformation. *Softw. Test., Verif. Reliab.*, 25(5-7):653–683, 2015.

[3] V. Aravantinos et al. AutoFOCUS 3: Tooling concepts for seamless, model-based development of embedded systems. In *Joint Proceedings of ACES-MB & WUCOR co-located with MoDELS*, 2015.

[4] F. Battiston, V. Nicosia, and V. Latora. Structural measures for multiplex networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 89(3), 2014.

[5] M. Berlingerio et al. Foundations of multidimensional network analysis. In *ASONAM*, pages 485–489, 2011.

[6] M. Berlingerio et al. Multidimensional networks: foundations of structural analysis. *World Wide Web*, 16(5-6):567–593, 2013.

[7] P. Bhattacharya, M. Iliofotou, I. Neamtiu, and M. Faloutsos. Graph-based analysis and prediction for software evolution. In *ICSE*, pages 419–429, 2012.

[8] P. Bródka, P. Kazienko, K. Musial, and K. Skibicki. Analysis of neighbourhoods in multi-layered dynamic social networks. *Int. J. Computational Intelligence Systems*, 5(3):582–596, 2012.

[9] P. Bródka, P. Stawiak, and P. Kazienko. Shortest path discovery in the multi-layered social network. In *ASONAM*, pages 497–501, 2011.

[10] F. Büttner, M. Egea, and J. Cabot. On Verifying ATL Transformations Using 'off-the-shelf' SMT Solvers. In *MODELS*, pages 432–448. Springer, 2012.

[11] A. Cardillo et al. Emergence of network features from multiplexity. *Scientific Reports*, 3, 2013.

[12] A. Cardillo et al. Modeling the multi-layer nature of the European Air Transport Network: Resilience and passengers re-scheduling under random failures. *European Physical Journal-Special Topics*, 215(1):23–33, 2013.

[13] J. Chimiak-Opoka. Measuring UML models using metrics defined in OCL within the SQUAM framework. In *MODELS*, pages 47–61, 2011.

[14] L. d. F. Costa et al. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. *Advances in Physics*, 60(3):329–412, 2011.

[15] S. Dorogovtsev and J. F. F. Mendes. Evolution of networks. *Adv. Phys.*, 51:1079–1187, 2002.

[16] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*. Wiley Publishing, 2008.

[17] E. Guerra, P. Díaz, and J. de Lara. Visual specification of metrics for domain specific visual languages. *Electr. Notes Theor. Comput. Sci.*, 211:99–110, 2008.

[18] J. Härtel, L. Härtel, and R. Lämmel. Test-data generation for Xtext. In *SLE*, pages 342–351, 2014.

[19] B. Izsó, Z. Szatmári, G. Bergmann, Á. Horváth, and I. Ráth. Towards precise metrics for predicting graph query performance. In *ASE*, pages 421–431, 2013.

[20] JaMoPP. *The Java Model Parser and Printer*, 2016. http://www.jamopp.org/index.php/JaMoPP.

[21] P. Kaluza, A. KÃűlzsch, M. Gastner, and B. Blasius. The complex network of global cargo ship movements. *Journal of the Royal Society Interface*, 7(48):1093–1103, 2010.

[22] P. Kazienko, K. Musial, and T. Kajdanowicz. Multidimensional social network in the social recommender system. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 41(4):746–759, 2011.

[23] M. KivelÃd', A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. Multilayer networks. *Journal of Complex Networks*, 2014.

[24] E. L. Lehmann and H. J. D'Abrera. *Nonparametrics: statistical methods based on ranks*. Springer NY, 2006.

[25] Y. Ma, K. He, and D. Du. A qualitative method for measuring the structural complexity of software systems based on complex networks. In *APSEC*, pages 257–263, 2005.

[26] Y. Ma, K. He, D. Du, J. Liu, and Y. Yan. A complexity metrics set for large-scale object-oriented software systems. In *CIT*, 2006.

[27] MOGENTES project. *Model-based Generation of Tests for Dependable Embedded Systems, 7th EU Framework Programme*, 2011.

[28] MONDO project. *Scalable Modeling and Model Management on the Cloud Project, 7th EU Framework Programme*, 2016.

[29] M. Monperrus et al. Measuring models. In *Model-Driven Software Development: Integrating Quality Assurance*. IDEA Group, 2008.

[30] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[31] V. Nicosia and V. Latora. Measuring and modeling correlations in multiplex networks. *Phys. Rev. E*, 92:032805, 2015.

[32] PolarSys. *Capella*. https://www.polarsys.org/capella/.

[33] J. D. Rocco et al. Mining metrics for understanding metamodel characteristics. In *MiSE*, pages 55–60. ACM, 2014.

[34] J. D. Rocco et al. Mining correlations of ATL model transformation and metamodel metrics. In *MiSE*, pages 54–59. IEEE, 2015.

[35] D. Steinberg et al. *EMF: Eclipse Modeling Framework 2.0*. Addison-Wesley Professional, 2nd edition, 2009.

[36] G. Szárnyas, O. Semeráth, I. Ráth, and D. Varró. The TTC 2015 Train Benchmark case for incremental model validation. In *TTC*, pages 129–141, 2015.

[37] L. Tang, X. Wang, and H. Liu. Community detection via heterogeneous interaction analysis. *Data Min. Knowl. Discov.*, 25(1):1–33, 2012.

[38] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.

[39] X. Wu et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[40] Yakindu. *Statechart Tools*. http://statecharts.org/.

[41] T. Zimmermann and N. Nagappan. Predicting defects using network analysis on dependency graphs. In *ICSE*, pages 531–540, 2008.